A Distributed Evolutionary Approach to Cooperative Vehicular Traffic Optimization

DISSERTATION
zur Erlangung des akademischen Grades

doctor rerum naturalium (Dr. rer. nat.)
im Fach Informatik

eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät
der Humboldt-Universität zu Berlin

von
Daniel Cagara

Präsident der Humboldt-Universität zu Berlin
Prof. Dr.-Ing. Dr. Sabine Kunst

Dekan der Mathematisch-Naturwissenschaftlichen Fakultät
Prof. Dr. Elmar Kulke

1. Gutachter: Prof. Dr. Björn Scheuermann
2. Gutachter: Prof. Dr. Ana Bazzan
3. Gutachter: Prof. Dr. Joachim Fischer

Tag der mündlichen Prüfung: 06.01.2017

Daniel Cagara, *A Distributed Evolutionary Approach to Cooperative Vehicular Traffic Optimization,* 2016

A Distributed Evolutionary Approach to Cooperative Vehicular Traffic Optimization

Daniel Cagara

ABSTRACT

The increasing amount of road traffic necessitates approaches that somehow "intelligently" organize traffic. In this context, the study of intelligent transportation systems (ITS) has been performed for some time. The goals of such systems include, e. g., is the dynamic optimization of route choices in a road network and hence the improvement of traffic conditions. There are two main methodologies how an optimization can be performed: the optimization towards a Nash equilibrium or towards a system optimum. While Nash equilibria can be easily reached, e.g., when every driver selfishly optimizes his own route, reaching the system optimum is a challenging task and requires all drivers to cooperate in an altruistic manner in favor of the system from a global perspective.

In this work, we discuss the design of a decentralized ITS that is capable of approximating system optimal route choices in the network avoiding that the drivers have to pay the full price of anarchy. The focus, in this context, lies on the applicability to real life situations where a number of difficulties has to be expected, e.g., an incomplete or incorrect view on the current traffic situation, the lack of future knowledge and an imperfect or limited communication channel. Facing these challenging questions, we develop solutions to a number of research questions, that arise from the aforementioned difficulties. Before we can do so, we focus on the fundamental concepts of traffic optimization with an emphasis both on the theoretical concepts as well as their applicability in real world environments.

One of the major contributions of thesis is an ITS design that performs an online optimization of all car's route choices. The idea is that, in a first step, cars use car-to-car communication to gradually learn about the current traffic situation. Then, each car works on a portion of one large instance of the route choice optimization problem. Our proposed approach utilizes past experienced conditions in the road network as well as information that has been broadcasted by other cars and utilizes it for leaning about the actual traffic situation. Car-to-car communication is used along the lines to coordinate the optimization process and synchronize the results between all cars. One major advantage is that our optimization in performed in real-time—while the cars drive—and is able to cope with sudden changes in the traffic dynamics and resource constraints in the communication channel. In this thesis, we evaluate the efficiency of our online optimization scheme for different types of traffic patterns and different penetration rates. We quantify the impact of imperfect knowledge on the optimization result and demonstrate the efficiency of our solution both in terms of travel time as well as in terms of other environmental variables such as the fuel consumption and the CO_2 emission.

ZUSAMMENFASSUNG

Durch ein zunehmendes Verkehrsaufkommen wächst die Notwendigkeit den Verkehr in irgendeiner Form "intelligent" zu organisieren. In diesem Kontext sind die sogenannten Intelligent Transportation Systems (ITS) in den Fokus der Forschung gerückt. Diese Systeme zielen in der Regel auf die dynamische Optimierung der Routenwahlen von Verkehrsteilnehmern ab und sollen dadurch die Effizienz des Verkehrs verbessern. Grundsätzlich kann die Vorstellung einer optimalen Routenwahl in eine von zwei Kategorien eingeteilt werden: das Nash Gleichgewicht und die systemoptimale Routenwahl. Während Nash Gleichgewichte vergleichsweise einfach erzielt werden können—beispielsweise dadurch, dass jeder Fahrer seine Route egoistisch optimiert—ist das Erreichen des Systemoptimums ungleich schwerer. Dieses setzt nämlich voraus, dass alle Fahrer miteinander kooperieren und gemeinsam eine Lösung finden, von der die Gesamtheit als solche profitiert.

In dieser Dissertation diskutieren wir das Design eines dezentralisierten ITS, welches in der Lage ist, eine systemoptimale Routenzuweisung im Straßennetzwerk zu approximieren, so dass die Fahrzeuge den price of anarchy nicht mehr in voller Höhe bezahlen müssen. Der besondere Fokus liegt hierbei auf der Anwendbarkeit des Ansatzes in realistischen Umgebungen, in denen eine Vielzahl von Schwierigkeiten zu erwarten ist. Dies beinhaltet beispielsweise eine unvollständige oder inkorrekte Sicht auf die aktuelle Verkehrssituation, das Fehlen von Wissen über Fahrzeuge, die erst in der Zukunft das Straßennetz betreten sowie ein nicht perfekter oder ressourcenlimitierter Kommunikationskanal. Um diese Fragen beantworten zu können entsteht eine umfassende Wissensgrundlage, welche sowohl die Modellierung der Dynamik des Straßenverkehrs beinhaltet als auch die Approximation einer systemoptimale Optimerung mittels verteilter, parallelisierter genetischer Algorithmen umfasst.

Zu den Ergebnissen dieser Arbeit zählen die Formulierung eines solchen ITS Designs, welches eine sogenannte Onlineoptimierung—also eine Optimierung zur Laufzeit, während die Fahrzeuge fahren—durchführen kann. Dabei verwendet der hier vorgestellte Ansatz nicht nur Informationen, die zuvor mittels Fahrzeug-zu-Fahrzeug Kommunikation verbreitet wurden—man könnte auch sagen von den anderen "gelernt wurden", sondern auch historische Erfahrungswerte über die "übliche" Last im Straßennetzwerk. Nachdem die aktuelle Verkehrslage den Fahrzeugen bekannt ist, arbeiten die Fahrzeuge jeweils unabhängig an einem kleinen Ausschnitt einer großen Instanz des Optimierungsproblems, welches die Routen aller Fahrzeuge im Straßennetz simultan optimiert. Auch hier wird Fahrzeug-zu-Fahrzeug-Kommunikation

eingesetzt, um die Koordinierung zwischen den Fahrzeugen zu ermöglichen. Der vorgestellte Ansatz kann mit den zuvor beschriebenen Schwierigkeiten von realistischen Umgebungen umgehen und reagiert auf die sich über die Zeit verändernden Verkehrsmuster im Straßennetz.

In dieser Dissertation evaluieren wir die Effizienz unseres vorgeschlagenen Ansatzes für verschiedene Verkehrsmuster sowie für verschiedene Ausstattungsdichten. Darüber hinaus quantifizieren wir den Einfluss von nicht perfektem oder nicht vollständigem Wissen auf die Qualität der Optimierung und demonstrieren die Effizienz des Ansatzes im Hinblick auf die Fahrzeiten sowie den Schadstoffausstoß/Spritverbrauch der Verkehrsteilnehmer.

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere gratitude to Professor. Dr. Björn Scheuermann. During my Ph.D. pursuit you were more than an excellent advisor to me. Thanks for your patience, motivation, and immense knowledge which inspired me always. I really appreciate that you always understood and tolerated that my optimistic "will be ready in two days" sometimes turned out to be two weeks or even two months. Thank you so much for giving me that freedom and trust.

I also wish to thank Professor Dr. Ana Bazzan. Ana, thank you so much for your brilliant comments and suggestions when we were writing our papers, your advice always has been priceless. Furthermore, I am especially grateful for the opportunity that I could spend a wonderful time in Porto Alegre under your guidance.

I have been given the chance to do my research in the context of two graduate programs: the Graduiertenkolleg METRIK and later BIGS^2. The interdisciplinary character that my research had as a direct result made my Ph.D. experience even more interesting and productive. Thank you very much for that opportunity.

Special thanks also go to my entire research group. You have been a source of friendships as well as good advice and collaboration at all times. I will miss you all, specially the time we spend together in the little "library" (which actually does have a few books by now) or at our Weisswurst events.

I would also like to acknowledge Florian Tschorsch: it was a pleasure to share an office with you during the last years. I very much appreciated all the discussions that we had on each other's research topics but I also enjoyed all the other smalltalk about all the other trending topics in a nerd's every day life.

Also, I want to thank my patient and understanding girlfriend Manon Clasen, who has put up with this foray into research from the beginning. You always encouraged me to stay focused, especially during the final, critical months of my dissertation.

Last but certainly not least, I would like to thank my family. Words cannot express how grateful I am to my mother Barbara Schneider, and father Werner Schneider for all your love and encouragement as well as the sacrifices that you've made on my behalf. Without you, I would not be where I am right now.

Thank you.

CONTENTS

1 INTRODUCTION

1.1 MOTIVATION AND PROBLEM STATEMENT

Traffic congestion is an urgent problem in most major cities and continues to increase with population and growth of urban areas. The trigger often is a severe oversaturation of the road network, especially during peak hours. More precisely, exceeding the nominal maximum capacity of a road segment, even just for a short moment, can drive the system into a jammed condition. Such a jammed condition can then take a significantly longer time to clear up again [32], deteriorating the traffic situation far beyond the original cause. The resulting delays add up to huge costs for society and business as well as to huge environmental impacts in the form of air pollution and fuel consumption. According to the Urban Mobility Report 2015 [93], the total amount of time that drivers spent stuck in rush-hour traffic was about 6.9 billion hours. The additional fuel consumption and the loss of time amounts to a total congestion cost of 160 billion dollars in the USA alone.

A reduction of congestion by simply expanding the infrastructure is quite expensive and often also impossible due to spatial limitations. However, it can be observed that congested traffic tends to concentrate on a few central spots, like major roads or large intersections during rush hour periods while most of the road network remains unchallenged. In this context, *Intelligent Transportation Systems* (ITS) [34] constitute one possible way to minimize the effects of congested traffic. The underlying idea is a more coordinated and "smarter" use of the existing road infrastructure. This makes these approaches more cost-efficient than other solutions [117]. More precisely, The term ITS is an umbrella for all kinds of techniques that make use of advanced communication technologies and real-time information in the context of transportation systems. That often means that multiple sources such as video cameras, induction loops, RFID readers or permanently installed radio-equipped stations—these are all stationary and referred to as road-side units (RSU)—are connected with radio-equipped vehicles. These vehicles, themselves, serve as mobile sensors and allow for an efficient and detailed collection of real time data. This data, in the basic form of ITS, then is used to provide drivers with the current traffic situation and traffic forecasts.

Advanced traveler information systems (ATIS) [8] are one subset of ITS. The goal of ATIS is to provide travelers with information that can be then used to pick decisions concerning route choice, departure

time, trip delay and the mode of transportation. However, there is no specification how ATIS have to look like.

Before the development of ATIS started, solutions usually only used static map data to compute the optimal path between a start and a destination point using a shortest path algorithm such as A* [51] or Dijkstra [33]. That means, these systems did not take the current (or future) traffic situation into account when providing the drivers with route suggestions. This quickly changed with the development and increase of traffic monitoring infrastructure.

Today's ATIS include solutions such as Google Maps [49], HERE Maps [54] and WAZE [105]. These approaches are not only making use of the information about the current traffic situation while searching for the best routes, they also incorporate historical data and perform traffic simulations in order to forecast the future traffic situation and improve the quality of the suggested routes.

Improving the quality of suggested routes corresponds to an optimization problem, where an arbitrary cost function is meant to be minimized. Nearly all ATIS, nowadays, are approximations of the so-called user or Wardrop equilibrium [103]; this is a variant of the Nash equilibrium and (in the context of road networks) means that a state is achieved, when no driver can individually benefit (in terms of his own cost function, e. g., the own travel time) from changing to an alternative route. Generally speaking, this equilibrium state is reached when each user tries to optimize his own cost in a selfish manner. However, as we discuss in this thesis, an equilibrium state does not necessarily reflect the best possible solution for the system from a global point of view. Due to the lack of cooperation, the equilibrium solution may have significantly higher costs (regardless of the actual cost function) compared to the global or system optimum where the costs for the system as a whole are minimized. This difference is often referred to as the *price of anarchy* [63, 87].

In this thesis, we answer the question how to build an ATIS which is capable of avoiding the price of anarchy by approximating a system optimal assignment of routes among the drivers. We cannot achieve this without cooperation between the drivers [9]. To our advantage, new advances in communication technology and the trend of the vehicular industry to equip more and more cars with communication-enabled devices have brought up a new type of ad-hoc networks: vehicular ad-hoc networks (VANETs) [116]. These networks usually use the IEEE Wave 802.11p standard [59] to allow effective communication between vehicles. However, as more and more cars also come with Internet-enabled devices on board, it is possible to use any other type of communication such as the mobile network (e.g., UMTS) to exchange data among the cars. Our overall goal is to exploit the capabilities of vehicular ad-hoc networks to implement a cooperative ATIS; the challenges of this idea will be discussed in the next section more thoroughly.

1.2 CHALLENGES AND CONTRIBUTIONS

In the remainder of this thesis we assume that the driver's route choices are the only variable that can be adjusted, i.e., the time of departure and the mode of transportation cannot be dynamically adjusted. The overall goal of this thesis is to present a decentralized methodology to approximate a system optimal assignment of route choices in a road network. Decentralization has many advantages: because there is no single point of control, there is no single point of failure and no authority which can restrict the use of the system or give certain participants (e. g., those who pay a fee) a prioritized treatment. Other advantages of our decentralized model, especially regarding the scalability of the system, will be outlined later. To summarize, the goal of this thesis is to provide the drivers with a free and open ATIS that—if the drivers cooperate—is capable of minimizing the impact of the price of anarchy and which scales itself with the growth of its user base.

As we are not interested in long-term behavior of traffic, we do not optimize macroscopically, that is, we do not look at cumulated traffic flows but rather individual route choices. Therefore, we conduct our analyses in microscopic settings, i. e., in settings where the movement of cars is not represented by a set of differential equations but rather by complex agent models which define their movement through the simultaneous interactions between all individual cars. In microscopic settings, user equilibria are reached as a direct result of non-cooperative games, that is when agents in a congested network choose their routes selfishly [44, 65]. Achieving the system optimal route assignment, on the contrary, becomes a more challenging task when the system is not described by a differentiable function [62].

This means that we do not have a continuous and differentiable problem suitable for traditional, mathematical optimization techniques. In fact, due to the complex nature of such microscopic agent based models, we are faced with a noisy, discontinuous, and non-differentiable optimization problem. In this thesis, we show that heuristic optimization approaches such as genetic algorithms (GAs) [55] are capable of providing very good results for this domain. One of the most important contributions of this thesis is the approximation of a system optimal route assignment online, i. e., meanwhile the drivers travel along their route. Facing numerous challenges, this in essence means that the approach functions without global knowledge including the lack of knowledge about the future and an imperfect knowledge about the present. At that point, this thesis separates from the previous studies.

In Chapter 2 we discuss the fundamental concepts of traffic and give an understanding of processes concerning the simulation and optimization of traffic. In this context, we discuss the so-called traf-

fic assignment problem which concerns the expected selection of routes between origins and destinations in transportation networks. Furthermore, we enlighten the properties of different traffic optimization techniques: we see the difference between equilibrium solutions where every driver optimizes his own route selfishly and solutions that can be found when all drivers cooperate. Besides that, we discuss different types of traffic simulations and the level of detail that they provide. Finally, we review and discuss existing solutions that address the traffic optimization problem and that have been proposed in literature so far.

In Chapter 3 we focus particularly on the methodology to optimize the route choices of individual cars in a road network using GAs under the premise of global knowledge. That is, the optimization algorithm has a-priori knowledge about all cars including their time of departure and their origin and destination points. While this approach is clearly based on unrealistic assumptions and so cannot be used for online traffic optimization (where challenges such as imperfect knowledge about the future or unpredictable driver's behavior have to be faced), it allows us to assess a road network's maximal optimization potential. Our contribution is to provide us with an idea about how far away the optimization result of existing approaches lies from the theoretical optimum. It is worth to emphasize that without a reliable upper bound it is impossible to assess the absolute quality of any traffic optimization approach. This makes this contribution essential for the remainder of this thesis.

GAs, in general, are a way to probabilistically search for optimal solutions to an optimization problem regarding a predefined target function. That is, the function that measures the quality of a solution and is meant to be either minimized or maximized. The underlying model gradually optimizes a set of solution candidates using methods borrowed from the evolution of living organisms. This process requires that the quality of such solution candidates is assessed in some way. In our context, this relates to evaluating the quality of different sets of route assignments in a road network by running a traffic simulation in order to predict the future development of the traffic situation. Whenever it is necessary to perform a large number of simulations, the traffic simulation becomes a bottleneck: depending on the size of the road network, such simulations may require a lot of computation time using today's state of the are traffic simulation tools that are often used in that domain such as the microscopic traffic simulator SUMO [10].

In fact, the time required for the large amount of simulations that a GA approach requires becomes an issue when trying to adapt the GA approach for online traffic optimization where time is a critical factor. We tackle this problem in Chapter 4, where we introduce a light-weight traffic simulation model. It uses a modified version of

the queuing model that was introduced by Gawron [44] to model the traffic dynamics during the microscopic traffic flow simulation. The general idea of this contribution is to make the simulation as light-weight as possible while at the same time retaining enough precision to be able to separate the good route assignments from bad route assignments. The contribution in this chapter is inevitable: due to the small computational footprint, our suggested traffic simulation scheme makes possible to perform larger-scale traffic optimizations that—as it is the case in our GA approach—typically suffer from scalability issues due to a huge number of required consecutive traffic simulation runs.

Chapter 5 combines the prior contributions. More precisely, it combines the GA approach with the fast queue-based simulation model to introduce a novel decentralized ATIS which is capable of assessing good route choices in dynamic (i.e., not predictable) environments by using a so-called island model GA. Island models [108] are one variant of parallel GAs. In island model GAs, each computation unit maintains its own GA instance (termed sub-population) which is locally optimized independently from the other sub-populations. In this particular case, we make use of the decentralized nature of our system and exploit the cars' navigation systems as computation units which work in consort towards a global system optimal solution. Such parallel implementations have often been shown to yield better overall results than using one large panmictic GA population [74, 108]. Such parallel implementations also imply that the sub-populations do not need to be in sync with each other. This is a huge advantage that allows us to distribute the optimization over many different entities which can be expected to run a large variety of different end user devices. Another contribution is the development of multiple techniques that handle certain problems that have to be expected in real world environments and which can be critical for obtaining a good result. This, to mention a few, includes imperfect knowledge about the actual traffic situation, the lack of future knowledge as well as an unreliable communication channel between the cars. Without paying special attention to these problems, its hardly possible for any optimization approach to get satisfying results outside of laboratory conditions.

Finally, we summarize and conclude this thesis in Chapter 6.

2 THE THEORY BEHIND TRAFFIC OPTIMIZATION

2.1 STATIC TRAFFIC ASSIGNMENT

Transportation systems are facing a variety of problems such as congestion and capacity constraints. In order to understand, reproduce and later mitigate these problems, it is important to model transportation systems as a whole by putting together its various components.

One important component constitutes the so-called traffic assignment. The starting point usually is an empty road network and empiric data about the demand in the road network. The demand is basically an estimation of how many travellers go from where to where and can be obtained by numerous ways such as observation, induction loops, floating car data, etc. The goal of (static) traffic assignment, now, is to determine the traffic flows for that road network which result from choices and interactions that occur in the population of the road network users described by the given demand. That is, the road network users make choices regarding which routes to chose to traverse from their origins to their destinations (the relations between origins and destinations are also commonly referred to as OD pairs). Of course, these choices result in several kinds of interactions, which then cause important phenomena such as congestion. To model these phenomena, some assumptions must be made regarding the pattern by which users typically choose their routes. In general, such pattern is supposed to comply with a performance criterion. This criterion usually involves some measure of disutility which is meant to be minimized. The problem of determining flows through the road network fulfilling the travel demands and such performance criterion is commonly referred to as the traffic assignment problem.

Typically, the two optimality principles of Wardrop [104] constitute two types of performance criteria which are commonly used for the traffic assignment problem. The first principle states that "*no driver can unilaterally reduce his/her travel costs by shifting to another route.*" [104]. More precisely, this principle is based on the assumption of a rational traveler. In this context, the interpretation of the "ideal solution" is that every user has found a route with the least travel time or minimum costs so that he has no incentive to change the route anymore for his own benefit. That is, each driver non-cooperatively seeks to minimize his own cost: the user equilibrium is then reached when no user may lower his costs by changing to a different strategy or route. This situation constitutes the so-called user equilibrium (UE), which, more generally, is a variant of the Nash equilibrium [46].

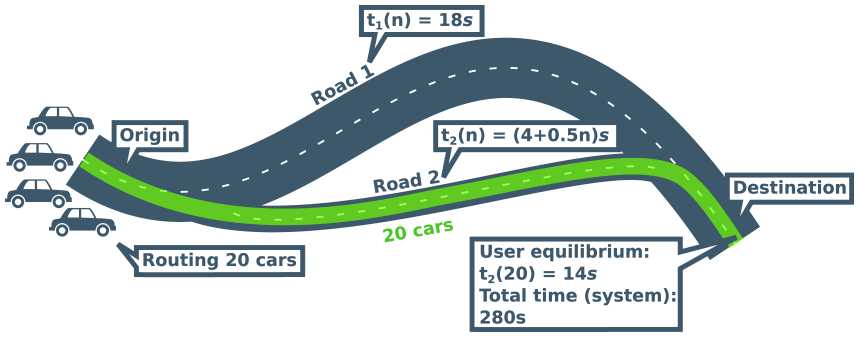


Figure 1: User equilibrium route assignment in a simple road network.

We want to take a look at a simple example to make this clear: Figure 1 shows a simple road network that consists of two alternative routes between one origin and one destination point. For a traffic demand of n, the link costs for the major road $road_1$ are fixed $t_1(n) = 18s$ and for the minor road $road_2$ the costs are denoted by $t_2(n) = (4 + \frac{1}{2} \cdot n)s$. Assuming that 20 cars are to be routed: in the user equilibrium case all 20 cars take the minor road. Wardrop's first principle is fulfilled: all cars have costs $t_2(20) = 14$ and no driver can improve his own situation by switching to an alternative route. In this particular case, that would result in changing the costs from 14 on the minor road to 18 on the major road.

However, the performance criterion can as well be formulated from the point of view of the system, i.e., using the total travel time of all travelers as the disutility. In essence, this reflects the so-called system optimal (SO) traffic assignment as it could be for example achieved not by a selfish traveler but by a central traffic control center. The system optimal traffic assignment is based on Wardrop's second principle which states that "*drivers cooperate with one another in order to minimize total system travel time.*" [104] This assignment can be also thought of as a model in which the costs for the whole system (e.g., the sum of all individual travel times) is minimized when drivers are told which routes to use regardless whether they profit or take a disadvantage compared to the case in which they optimize the route selfishly for themselves.

Figure 2 shows the system optimal traffic assignment in the same simple example scenario that we have used before. It can be seen that, albeit the majority of 15 cars could improve their travel time from 14s (compared to the user equilibrium case) to 11.5s, a small number of cars has to accept a disadvantage: they need 18s which is four seconds longer than the 14s in the user equilibrium case.

It can be seen that there are cars in the system optimum case who could improve their own travel time by switching to a different route but altruistically refrain from it in the favor of the system as a whole. That is, in the given example, the sum of all travel times is minimal. More precisely, the total system costs for the SO case are $280 - 262.5 =$

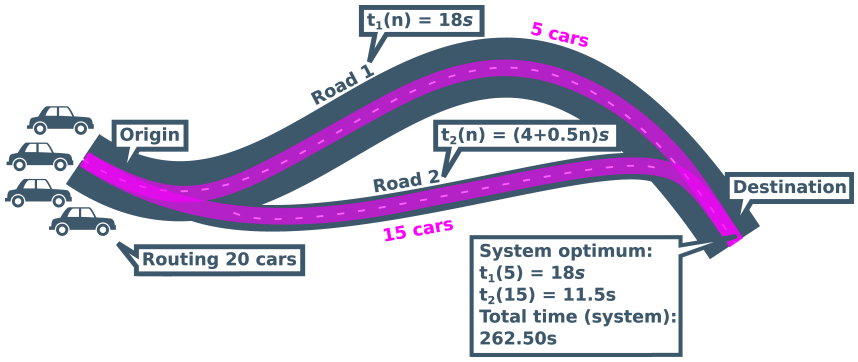


Figure 2: System optimal route assignment in a simple road network.

22.5s lower than in the UE case. This difference, that is, the difference between selfish routing and a cooperative solution, is referred to as the *price of anarchy* [87] which is a measure of how inefficient the Nash equilibrium of a game can be compared to a system optimal solution [87].

When interpreting the traffic movement as a flow, i. e., in the macroscopic case, the UE and the system optimum solutions can be numerically solved by the Frank-Wolfe algorithm [41]. This algorithm can solve optimization problems where the objective function is convex and the constraints of the problem are linear. This algorithm converges to an exact solution. However, a disadvantage of the Frank-Wolfe algorithm is its speed of convergence, which is too slow for most practical uses, even considering acceleration methods that have been proposed in [17, 44].

When cars are treated as individual entities, i. e., when we inspect the microscopic case, this looks quite differently. The UE is reached when all cars who continuously switch to their best alternative reach a state in which no driver can unilatery increase his travel time anymore. The system optimum, on the contrary, can only be found by trying out all possible route assignments that are possible (and evaluate them with a microscopic traffic simulator). As this is computationally not feasible, greedy or heuristic optimization approaches are often used here to obtain a solution that is (hopefully) very close to the system optimum.

2.2 DYNAMIC TRAFFIC ASSIGNMENT

Static traffic assignment is a useful concept when the modeling horizon is long enough, as for instance for long-term planning or for the evalution of the impact of future changes in the demographics on the overall performance of a transportation system. However, and we only consider the case of finding a Nash equlibrium here, static traffic assignment always assumes the presence of one particular steady state. Unless a long time horizon is considered, however, steady state

virtually inexist in traffic networks. This renders the static traffic assignment ineffective for traffic management. A solution is proposed by an alternative traffic assignment method: the so-called *dynamic traffic assignment* (DTA), which, in contrast, accounts for the dynamic properties of road traffic such as, e. g., the variation (over time) in travel times on links [27]. To put it differently, in contrast to the static variant travelers now are assumed to know *future* travel conditions along the journey between an OD pair. And as a result of this, they aim at minimizing the travel time considering the condition of a link at the moment in which they will actually use it which again depends on when they arrive at the other previous links along a route. Hence, travelers who depart from an origin to a destination at different times will experience different travel times: a user equilibrium condition here only applies to travelers who depart at the same time between the same OD pair.

DTA, in this context, requires for a methodology to allow for finding routes with the least costs considering that the link travel times change over time. This can be achieved by so-called time-dependent shortest path (TDSP) algorithms [80], which minimize the actual experienced travel time. What follows is an iterative algorithm which consists of three phases. The algorithm starts with some initial set of route choices and aims for gradually improving them. This is usually achieved in three phases: a phase in which the effects of all route choices on a global cost function are determined, a second phase where the routes with the lowest experienced travel times (that is, for each OD pair and assignment interval) are determined, and a third phase where adjustments are made in the assignment from the second phase [27]. In the last step, just a fraction is allowed to change their route to avoid a number of side-effects such as oscillations due to too many users shifting to an alternative route at the same time. Applying this three-phases over and over again successively brings the system closer to an equilibrium solution until a convergence criterion is met. The Nash equilibrium obtained by this method, in this context, would be referred to as the *dynamic user equilibrium* (DUE) [27].

We have only discussed the Nash equilibrium here, because the system optimal route assignment obtained from the static traffic assignment already accounts for the temporal variation of the link costs.

2.3 AGENT BASED TRAFFIC OPTIMIZATION

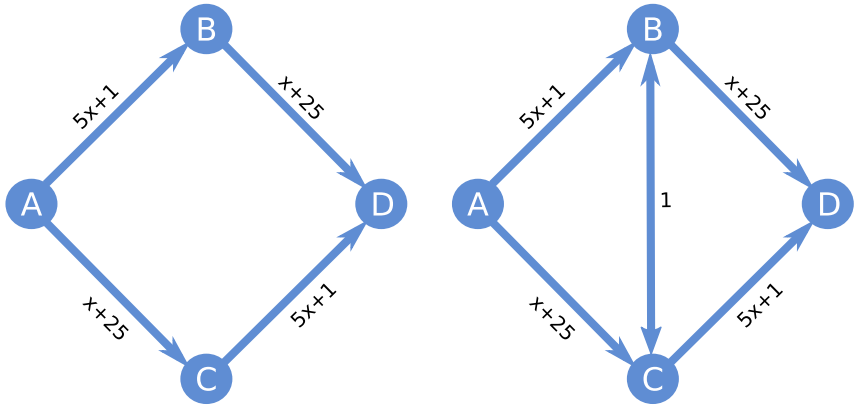
The former two traffic assignment methods are of theoretical nature and allow for a traffic assignment based on global knowledge, that is, knowledge about the past, the present and the future. They are perfectly fine when historical data is used to draw conclusions about a system that can be fully described. This, of course, cannot be assumed

in systems that aim to work "live", e. g., in a navigation system that is meant to assign good route choices to all equipped cars when only the past and the present is known (and not the future). Static and dynamic traffic assignment methods reach their limits here.

Hence, the creation of (co-ordinated) navigation systems necessitates good approaches to embed "traffic assignment" into the context of on-line navigation. One way to achieve this is to reuse the ideas from DTA in order to optimize traffic conditions "online" using so-called agent-based approaches. Those agents then continuously perform either dynamic traffic assignment or system optimal static traffic assignment based on the current traffic situation. The most important difference to the two prior approaches is that if the traffic situation changes unpredictably, the agents can only adapt their future behaviour and not their earlier decisions.

More precisely, agent-based approaches, every driver is represented by a so-called agent. Each agent is a discrete entity in the model and is associated with certain attributes (e.g., type of car, maximum speed, aggressiveness, etc.). Furthermore, agents might be restricted by the availability of required resources such as, e.g., the knowledge about the (exact) current or future traffic situation. To overcome such restrictions, agents have the capability to communicate and cooperate with other agents. When doing so, they can, e. g., solve problems together or coordinate the route choices among each other. Furthermore, in an agent-based approach, each agent in principle can decide autonomously whether or not he adapts his route. Thus, the key element in the classical DTA approach, namely, the ability to determine an arbitrary fraction of travelers to change routes, is no longer possible: although these agents are equipped with devices that enlarge their knowledge of the congestion status of the whole network, they still may be able to perform experimentation that the classical DTA approach does not necessary foresees. This can be the case for both the selfish optimization of the own route as well as in the case of a centralized route guidance system. These are the two key differences to (and at the same time the limitations of) the original idea of DTA.

Let us go back to the agent based approaches: each agent is associated with a route choice model which assigns each driver with a route from his origin to his destination destination point. Route choice models are either based on a centralized paradigm or on all drivers selecting their routes individually. In the former some central authority can choose the route of every driver and so aim for optimizing a global cost function, e.g., the sum of all travel times. In the latter, the driver is usually modelled according to the assumption of a rational traveller who aims at optimizing his own route selfishly. The idea behind agent based approaches is, that each agent is modeled as an autonomous entity with the aforementioned decision making



(a) An exemplary four-link network.

(b) Same network with highspeed link added.

Figure 3: Illustration of the braess paradox.

heuristic, with a degree of uncertainty and the ability to learn and/or adapt his behavior to external conditions. The overall observation of the system, such as the observation of the traffic dynamics, is then a direct result of the simultaneous actions and interactions between these autonomous agents.

Whenever a new kind of navigation or route guidance system is developed and meant to be evaluated in terms of performance under conditions that reflect the reality as closely as possible, a so-called multi-agent simulation is typically used.

2.4 BRAESS PARADOX

The general problem with agent based approaches that do not involve a cooperation between the drivers and rather chose their routes individually is that multiple drivers can simultaneously select one common "best" route and create a situation as in the Braess paradox.

The Braess paradox, generally speaking, was first discovered by Dietric Braess in 1968 [15] who discovered that adding route choices to a network can sometimes increase the costs of all driver in the road network. Figure 3a illustrates a simple four node network. The nodes are connected with edges that in all cases have a linear cost function (referring to the time required to traverse that edge in minutes) depending on the edges' current traffic load x. There are two possible routes between nodes A and D: $r_1 = (A, B, D)$ and $r_2 = (A, C, D)$. Both routes, in this context, have the same cost which is $\text{cost}(r_{1,2}, x) = 6 \cdot x + 26\text{m}$. As both routes are equally attrative, it can be expected that traffic will split across the two alternatives equally. In the case of 6 drivers, this would relate to 3 of them taking each route with a travel time of $\text{cost}(r_{1,2}, 3) = 6 \cdot 3 + 26\text{m} = 44\text{m}$.

Now, let us apply a small modification to the road network. As depicted in Figure 3b, we have added a high speed link between nodes

B and C which has the static cost of 1m. To begin with, we are performing the route choices for all travellers successively. For the first of our 6 travellers, the best option to travel from A to D is taking the route $r_3 = (A, B, C, D)$ with a travel time of 13m. The second, third and fourth travellers take the same route with a travel time of 23m, 33m and 43m. This, again, leaves $r_1 = (A, B, D)$ and $r_2 = (A, C, D)$ as the best routes for the last two travellers. However, those two travellers now have a best possible travel time of 52m which is around 20% worst compared to the situation in the original road network.

It can be clearly seen, that the additional edge only had a positive effect on the first travellers while, after all, harming the system as a whole. This effect may become even bigger when travellers neither coordinate themselves nor have the knowledge about the most recent situation in the road network.

2.5 TRAFFIC SIMULATORS

We have seen, that both the dynamic traffic assignment as well as the agent based approach to traffic optimization relies on traffic simulations. In general, a traffic simulation (model) is a mathematical or algorithmic representation of a virtual dynamic system (e .g., a model of a small city) that can be used to draw conclusions about the properties of the real world. Numerous traffic simulation models have been proposed in literature of which the most important ones will be outlined here. Depending on the level of detail that is required, these models can be categorized into *macroscopic traffic models* and *microscopic traffic models*.

Macroscopic traffic flow models are basically models that mathematically formulate the relationships between traffic density, traffic flow, and average traffic speed. The general idea behind macroscopic traffic flows is the assumption, that traffic flows—when looked at from a distant on-top view—behave similar to fluid streams. One of the first approaches to model traffic flow at a macroscopic level was presented by Lightwill and Whitham after they discovered that *traffic flow on long crowded roads* and *flood movements in long rivers* behave somewhat similar [67] and—one year later—published their idea of the so-called LWR model [85] which, after numerous iterations, is today known as the Fundamental Diagram [31]. Macroscopic models only reflect how the traffic *as a whole* behaves. This can be useful for simulating large scale real-world simulations and inspect long term effects, e.g., effects of adjustments in the road network structure. More precisely, macroscopic traffic models are often used to evaluate so-called equilibria (cf. Section 2.1)—these are stable traffic states which attune after a sufficiently large period of time has passed [73] [44].

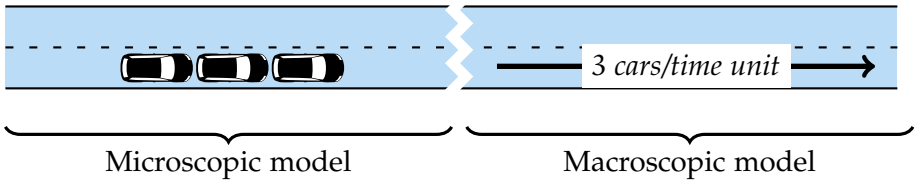


Figure 4: Difference between micro- and macroscopic traffic models.

Yet, sometimes it might be useful to have a more detailed look. In contrast to macroscopic traffic models, which allow for a fast and efficient simulation, the so-called *microscopic traffic models* are meant to model traffic dynamics in more detail (cf. Figure 4). That is, traffic is described at the level of individual vehicles. That means that the vehicles' actions as well as the interaction of the vehicles among each other are modeled in a way so that they represent what can be observed in reality as accurate as possible. Essentially, the vehicles' behavior can be described by three models: a car-following model, a lane-change model, and a route choice model. The car-following model describes the breaking and accelerating patterns that result from interaction of the driver with the vehicle in front as well as other objects (such as speed limits, road curvature, etc.). The lane-changing model describes the decisions when to change lanes, based on the driver's preferences and the situation in both the current lane and other lanes (speed of vehicle in front, sufficiently large gap in adjacent lane, etc.) The route choice model describes how drivers determine which path to take from their starting location (origin) to their destination, and how they react to traffic and route information along the way from their current position to their desired destination.

Microscopic traffic models are often used to evaluate situations in which the behavior of individual vehicles is adjusted or observed—such as in (cooperative) navigation systems. Also, certain effects, e.g. effects caused by traffic lights, can only be reproduced using microscopic traffic models.

Mesoscopic models fall in between the former two models: here, we still have individual vehicles that move across the road network, yet, their movement is modeled on a macroscopic basis.

There are numerous traffic simulators that rely on microscopic traffic models. SUMO [10], for example, is an open source, fully featured microscopic traffic simulation framework. It offers a very detailed level of simulation, where the behavior of all individual cars in the scenario is simulated using a set of configurable and exchangeable driver models. Furthermore, it comes equipped with a large amount of different analysis tools which handle tasks such as (user equilibrium) route optimization, visualization, emission calculation, etc.

In the remainder of this thesis, we use SUMO to simulate the real world (consisting of the road network as well as the movement and behavior of all traffic participants) as closely as possible. SUMO is based on a number of models which reflect the state of the art knowledge in traffic engineering. Still, every model to some extent relies on simplified assumptions. When interpreting the experimental results, hence, it is always necessary to consider the validity conditions of the underlying models as well as their simplifications and assumptions. Whenever we refer to the applicability to the real world, we effectively mean the applicability to the real world as it is described by the underlying models.

When simulating scenarios with a very large number of network participants, typical microscopic approaches become quickly reach their limits. Applications that are time critical as, e.g., navigation systems that perform an online optimization of route choices and which rely on traffic simulations on the microscopic level require for faster ways to perform—or at least to approximate—microscopic simulations.

MATSim [25] assesses this "scalability" problem by focusing on a very fast, graph-based simulation specifically tailored for large multi-agent simulations. It uses the queuing model introduced by Gawron [44] to model the traffic dynamics during the microscopic traffic flow simulation. To point out, even when the detail of the individual car's movement is reduced significantly, MATSim is still considered a microscopic simulator as it retains the granularity of individual vehicle movements.

One important aspect of microscopic traffic simulations is the modeling the the dynamics inside intersections, where multiple flows compete with each other. The intersection logic by Gawron works in a way that all links are processed in an arbitrary but fixed sequence and a vehicle is moved to the next link if a) it has arrived at the end of the link; b) it can be moved according to capacity; and c) there is space on the destination link. The problem with the Gawron's queue model is that links are always selected in the same sequence, thus giving some links a higher priority than others under congested conditions. MATSim conquers this problem with the introduction of "fair intersections". Those intersections eliminate the queues on the links, and replace them by queues in the intersections. This corresponds to a demand-based balancing. This unfortunately is not very realistic, when taking a look at the different junction dynamics that can be observed, for example, in the SUMO simulator. This approach, to give one example, does not properly handle locked flows that may occur as a result of priority lanes or right-before-left-situations. One of the contributions in this thesis will address this issue.

2.6 PRACTICAL APPROACHES TO TRAFFIC OPTIMIZATION

When taking a look at the various publications in the domain of traffic optimization techniques, one can distinguish two main directions of research. A first category focuses on the fundamental question how to distribute traffic in the road network in order to achieve an optimal solution. Optimal solutions are in this context defined by minimizing a target function; typically the travel time is used as the target function but others, as we will see later on, possible as well. These approaches usually assume ideal conditions and suggest theoretical concepts of how solutions to the minimization problem could look like. The second category concentrates on tailoring those theoretical principles for the practical use in real-world environments. In this context, numerous problems expected in real-world environments have to be tackled, such as imperfect knowledge about the traffic situation or unpredictable driver behavior.

In the majority of cases this relates to integrating solutions, which are capable of dealing with the limitations expected in real-world environments such as imperfect knowledge, unpredictable drivers, and a limited communication channel, into practicably usable traffic information systems.

We first want to take a look at the former category of approaches. The authors of [91] demonstrate the applicability of GAs to traffic route optimization. Specifically, they show that their approach offers clear advantages over other approaches such as commercially available nonlinear programming software which have been typically used back then [118]. However, they use a simplified (macroscopic) traffic model, which does not reflect the complex dynamics observed in real-world traffic. Furthermore, they only optimize the exiting flow at one exit link in a very simplified road network which does not take care of effects such as feedback loops when many crossing / competing flows to many exit links occur. Despite its simplicity, this work has introduced a novel idea which has been picked up by many different researchers in the future.

One—pretty sophisticated—way to apply GAs to traffic route optimization has been presented in [95]. Contrary to the macroscopic approach mentioned above, the authors propose an approach to optimize traffic flow using a GA in a microscopic setting. This approach is targeted to practical use in a real world traffic information system by periodically repeating short-term forecasts of the traffic situation.

A different approach is presented in [72] and uses statistical demand data to estimate the traffic load. That in essence means, that some sort of empirical knowledge about how traffic patterns usually look like at different points in time has to be present. Based on this estimation, an approximation of the optimal route assignment is searched using GAs. The optimization in this contribution is based

on past knowledge, but does not incorporate the changing conditions in the road network during the optimization in the same manner as we want to do. That means, this approach a-posteriori searches for a route assignment, that would turn out to be good given a particular *past* traffic situation. This approach is however problematic: when applying a route choice optimization to the scenario, the empirical data about how the traffic usually looks like in that particular scenario, will—for obvious reasons—no longer be valid. After changing the behavior of the agents, the empirical data would have to be collected again to ensure a further optimization (e.g., when new agents enter the road network after the optimization has been performed). To put it different, the bidirectional feedback loop between route choices and the resulting traffic situation is not considered here. Therefore it is tailored to assess good route choices for a given equilibrium state of a network, yet does not adapt to highly dynamic changed in traffic condition such as observed during peak traffic hours. This again demonstrates the importance of the elimination of required global knowledge for the prediction time span: this is one of the core problems that we aim to solve in the remainder of this thesis. Also, a downside of this approach is that the authors use macroscopic modeling only. This means that cumulative flows are optimized instead of the route choices of individual vehicles.

The so far mentioned selection of approaches concentrates on the fundamental idea about how route choices and traffic flow can be optimized. Often they assume ideal conditions (e. g., a global view on the road network or valid empirical knowledge about the expected traffic situation), which are not given in a real-world environment and/or ignore the co-dependencies and feedback loops which are to be expected in real traffic environments. However, road networks cannot be always expected to be in an equilibrium state and the traffic situation (be it empirical or current) has to be *learned about* first. Furthermore, in real systems it cannot always be expected that drivers behave exactly as they are told to. These issues of real-world applicability, which also include the communication between the vehicles and optimization based on incompletely learned knowledge have been also assessed the literature.

One of the first approaches to deploy a traffic information system for real world use with the intention to optimize traffic conditions was INRIX [92]. Their approach records the current traffic condition in a central database. Drivers can then collect the estimated travel times for multiple routes from a central database in order to select the best one. The general problem with such approaches is that multiple drivers can select one common "best" route and create a situation as in the Braess paradox [40]. Similar approaches based on information exchange and independent local decisions are discussed in [29, 61, 89, 90]. Picking a route using such approaches is also referred to as Dy-

namic Shortest Paths (DSP). The impact of the lack of a feedback loop – a car's route choice is not taken into account in other cars' choices and vice versa – has been already assessed by the community. Elementary techniques to overcome the resulting problems, such as the A* shortest path with repulsion (AR*), the random k shortest paths (RkSP), the entropy-balanced kSP (EBkSP), and the flow-balanced kSP (FBkSP), are described in [82]. These approaches can be easily integrated with one of the traffic information systems described above. A similar approach is presented in [77] and was evaluated using a state-of-the-art traffic simulator. While simulations show the effectiveness of these methods, they do not strictly reflect a system optimal route assignment.

A similar approach to [95] was followed by [79], which is essentially a real-world application of the work presented in [1]. Here a combination of car-to-car and car-to-infrastructure communication is used to gather information about the current traffic situation. Then a continuously running GA instance is monitoring attentively the traffic situation and reacting by (partial) detours around congested areas. The key difference to what we want to achieve is that the authors let the cars optimize the route choice locally just for themselves, while we aim for a *joint*, distributed optimization towards a system optimum using the local navigation systems as cooperative computation devices. To recall, the selfish route optimization always correlates with the requirement to pay the price of anarchy [88].

GAs are however not the only possibility to assess good route choices. In [42], the authors' goal is again to approximate an optimal distribution of traffic in road networks. More precisely, the authors show that game-theoretic approaches, where drivers choose their routes independently, can be applied to the route assignment problem as well. So do the authors of [7]. There, a game with three populations is defined where good solutions are "learned" through reinforcement and replicator dynamics [14]. Other approaches following a similar pattern include traffic optimization derived from the behavior of ant colonies [36]. However, game-theoretic approaches generally target equilibrium solutions [30] which are often problematic in highly dynamic traffic networks. In this thesis we, on the contrary, specifically aim for system optimal routes. Compared to an equilibrium solution this again avoids the price of anarchy.

Another approach to road congestion minimization is presented in [18]. The idea is to tackle the problem from another direction. Instead of suggesting the drivers with alternative routes, they use a biased random-key GA to balance the load in a road network by strategically allocating tolls/fees on particular links. In contrast to this solution, which looks very promising, this thesis focuses on traffic optimization by optimizing *route choices* and not applying any changes to the road network or its structure.

In [57], a mathematical approach to optimizing traffic from the systems perspective is pursued. Again, due the complexity of the problem, several abstractions are made—for instance, the lack of dynamic traffic flows precludes the application to real-world traffic situations. Although they work with static traffic flows, their future work incorporates taking the dynamics of traffic into account. Our goal is specifically to introduce a GA based optimization strategy, in which we are able to avoid such simplifications.

The existing body of work on traffic assignment in a more general sense includes, for example, approaches like [60], where an ant-hierarchical fuzzy system is applied.

Also, there are several approaches that have already found adoption in the read world. These approaches, to mention a few examples, include Google Maps [49] and WAZE [105]. These approaches follow the general idea of a traffic information system: floating car data is collected and used to estimate the travel time on the links in the road network. This information is then used by the navigation systems to determine the quickest route from a source to a destination. Other similar approaches such as [43, 94] pursue the same approach while sharing additional data such as fuel consumption or drivers experience in order to allow the navigators a multi-objective route optimization. Here, again, coupled (uncoordinated) route choices may lead to suboptimal results, an issue that has been addressed by numerous papers pursuing cooperative routing (i. e., sharing also the individual route choices) [52, 112, 114]. The focus of these papers lies on multiple difficulties of cooperative navigation systems such as the real time acquisition of the necessary information. These approaches, however, do not aim for a globally optimal solution, but rather optimize the local user's route choice based on the information that has been gathered at this time. The authors of [68] pursue a comparable approach, however they collect trajectories and routing decisions from all cars in order to balance the route choices centrally at a server using the EBkSP algorithm. That is, it computes k shortest paths for an OD pair and gives the user the path with the least expected popularity.

In contrast to most of the presented approaches, we specifically aim for creating a traffic information system that is capable of assigning system optimal route choices among the drivers and does not rely on a central authority as well as works—contrary to the solutions that aim for an equilibrium solution—in highly dynamic setting with quickly changing traffic patterns. This requires two steps. In a first step we will on purpose avoid producing a system that is able to work in a dynamic real-world scenario and so we at first do not need to care about the downside of an incomplete view on the traffic situation nor do we need to expect that the drivers behave differently than expected. Rather, the first methodology we will propose aims for approximating an optimal route distribution in a specific scenario from

an on-top (that is, assuming global knowledge about the present, past and the future) view on the network. This will give us an idea about the optimization potentials of different road networks. Our approximation of an optimal route distribution then can be used to compare against heuristic approaches that work with local knowledge only, and will give us an idea about how good the route choices computed by them actually are. In a next step we will adapt this methodology to work in dynamic real-world environments with all the associated pitfalls and problems.

3

ROUTE CHOICE OPTIMIZATION USING GENETIC ALGORITHMS

3.1 CHAPTER OVERVIEW

This chapter is based on previous work by the author [20, 21] and reflects his contributions to the respective work.

In this chapter we introduce a methodology to optimize the route choices of individual cars in a road network using GAs. GAs are heuristic search methods, which are suitable to solve complex optimization problems. We cover GAs thoroughly in Section 3.2.

The focus of this chapter lies on a methodology that uses GAs and that is capable of approximating a system optimal route assignment in a road network (cf. Section 2.3). At this stage, we specifically assume global knowledge about which cars are currently present in the road network as well as a perfect knowledge about what cars (including their origin and destination points) will enter the road network in the future. We are aware that these assumptions prevent this system to be productively used in real world environments—this is not the goal here. At this point, we are interested in the overall optimization potential that one specific traffic scenario has. This later allows us for comparing both existing and future ITS with regard to how close these solutions come to the system optimal route assignment.

Applying GAs to traffic optimization requires handling several pitfalls. For example, the design of the search space is a crucial factor that influences the speed of the GA's convergence towards a solution to the optimization problem. The search space is a set of all possible solutions which is given the algorithm as an input. Our challenge is twofold. On the one hand, our goal is to keep the search space as compact as possible, in order to increase the convergence speed of the GA. On the other hand, we have to model the search space in such a way that it very likely contains good solutions. Solutions that are not part of the search space will not be found by any optimization.

In this chapter, we introduce our methodology, evaluate its performance in a specific road network, and compare it to other traffic optimization algorithms discussed in literature. The diversity of the solution candidates of a GA is a crucial factor that has a great impact on the quality of the result. Hence, a diversity analysis is undertaken, which shows that the diversity of the population of solutions was ensured by our approach and thus the potential of GAs was fully exploited. This comparison not only shows that the proposed optimization algorithm leads to better results and thus shows that existing approaches do not exploit the whole optimization potential, but it also forms the basis for the remainder of this thesis. That is, giving us

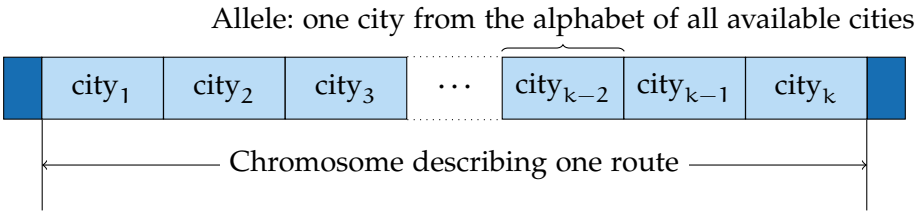


Figure 5: Illustration of chromosome for traveling salesman problem.

an idea of the overall optimization potential of arbitrary road traffic scenarios; this gives us an idea of the "best case" which then can be used to judge how close alternative approaches, e. g.approaches that do not assume global knowledge, can come to a theoretical optimum.

3.2 GENETIC ALGORITHMS

3.2.1 *Methodology*

First, we want to take a closer look at the principles behind GAs. Generally speaking, GAs are heuristic search methods based on the principles of the evolution of living organisms [47] and can often be used to quickly assess "good" solutions to complex, computationally unfeasible optimization problems. NP-hard problems such as the *system optimal route assignment problem* are a natural candidate for heuristic optimization approaches such as the here discussed GAs [55]. Generally, GAs search for a solution inside a "search space" that has to be defined a-priori. Each point in the search space is one possible solution candidate for the optimization problem. The design of the search space, in this context, is essential for the actual performance of the GA: good solutions that are not part of the search space thus cannot be found.

A drawback of genetic algorithms is that a solution's quality can only be seen in comparison to other known solutions; genetic algorithms actually have no concept of an "optimal solution," or any way to test whether a solution is optimal. In fact, testing for optimality is nearly impossible in large non-linear settings such as the route assignment in large, microscopic traffic simulations. Hence, we can only expect an approximation of the optimal solution.

Possible solutions to the optimization problem, also termed candidate solutions, are stored in finite-length strings of an alphabet and are referred to as chromosomes and the individual characters of the alphabet on the chromosome are called alleles. For example, when trying to solve the traveling salesman problem [53], the chromosome would represent a route and the alleles on the chromosome would describe which cities are to be visited in which order (cf. Figure 5).

To evolve good solutions, a methodology is required for distinguishing good solutions from bad solutions. In this context, we intro-

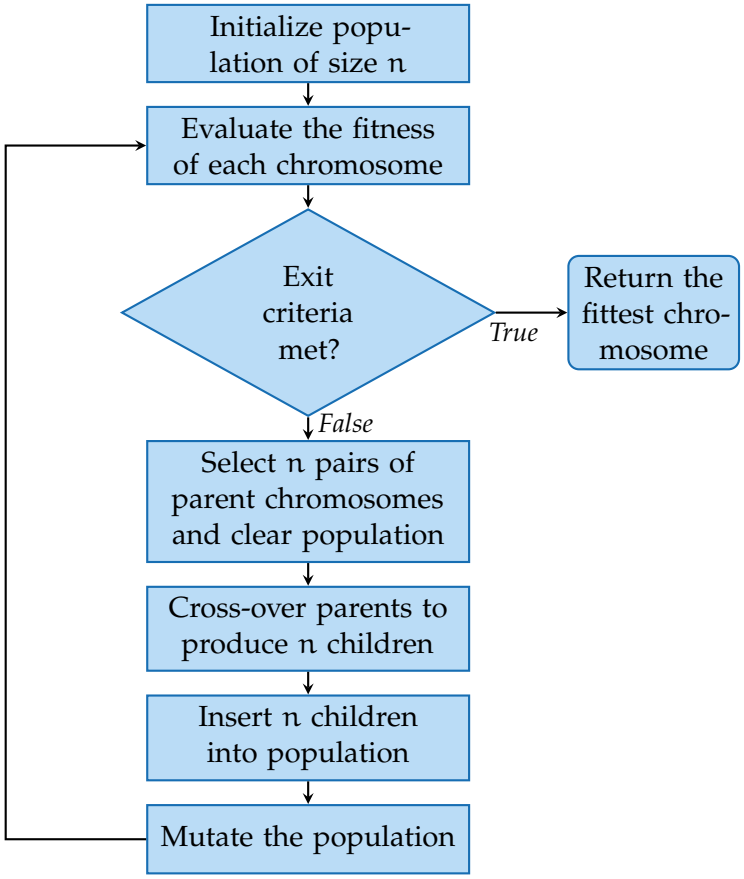


Figure 6: Flow chart of genetic algorithms.

duce a measure which is referred to as the *fitness function* and assigns a score, i.e., a *fitness* to each solution candidate. The *fitness value* of an individual can, e.g., be determined by a mathematical function as well as it can be determined by a computer simulation or any other methodology as long as it provides a reliable total ordering of solution candidates among each other based on their *quality*. Generally speaking, we consider the fitness function to be a function $f(x) = h$ that assigns a fitness value $h \geqslant 0$ to each chromosome x in the search space. In the same example as above—that is, the traveling salesman problem—this function returns the inverse of the total cost that is required to visit all cities in the order described on the chromosome.

The goal of the GA is to find $x_{max} = \underset{x_{cand} \in S}{\operatorname{argmax}} f(x_{cand})$, i. e., a chromosome (or solution candidate) x_{max} in search space S for which the *fitness value* is maximal. The general schema of GAs is illustrated in Figure 6: the starting point is an initial population of size n. In this context a population is a set of chromosomes—or solution candidates—which is generated randomly. The quality of this population—in the beginning, i. e., in generation $g = 0$—is poor.

In a next step, all individuals in the population are evaluated in terms of their fitness value. This allows for distinguishing the good

Figure 7: Illustration of the Roulette-Wheel Selection method.

individuals from the worse ones. Also, at this point the GA checks if the *exit criterion* has been met. The exit criterion can be defined arbitrarily. While commonly the GA is configured to exit when the fitness of the best individual in the population has stagnated, i.e., not changed for the last p generations by an amount greater than ϵ, it is also possible to exit the GA after a fixed number of generations or even let it run continuously with no exit criterion set. If the exit criterion is set, the GA returns the fittest individual in the population. Otherwise, the population is evolved into a subsequent generation and the process is repeated. The evolution involves three types of *genetic operators* which are discussed in detail in the following section.

3.2.2 *Selection Schemes*

After all individuals in the current population have been evaluated in terms of their fitness value, the so-called *selection operator* is applied. The selection operator is meant to select individuals from the population for reproduction. The general idea of the selection operator is to select individuals in a way that individuals with a higher fitness value are more likely to be selected. In this context, two main methods for choosing individuals for reproduction exist: the *fitness proportionate* and the *rank based* method [6].

The fitness proportionate selection method—also termed the Roulette-Wheel Selection or stochastic sampling with replacement — is the simplest selection routine. This method is based on a stochastic algorithm and corresponds to the following technique: in the beginning, the individuals are mapped to pairwise disjoint slices of a virtual roulette wheel in a way that the size of each individual's slice is proportional to its fitness (cf. Figure 7). Two random points on the roulette wheel are then selected and the individuals whose slices span the random point are selected to be the both parent individuals for the offspring.

The rank-based selection method works quite differently: the individuals in the population are sorted according to their fitness values. Here, the fitness value only determines the position of an individual (i.e., the *rank*) in the sorted set. Its absolute value is not taken into account in the selection process.

Starting with the sorted set of individuals, selection can now be performed using either the *linear ranking* or the *non-linear* ranking method [83]. Let n denote the number of individuals in the population and p_x with $1 \leqslant x \leqslant n$ the position of an arbitrary individual x in the sorted set while the worst individual is found at position $p_x = 1$ and the best individual is at position $p_x = n$. Furthermore, let S be the selective pressure, i.e., the probability of the best individual to be selected compared to the average probability of selection of all individuals. Before selection, each individual is assigned with a *virtual fitness value* f_{vir} which is calculated from the individuals position in the sorted set. According to [109], the virtual fitness can be calculated as

$$f_{vir}(p_x) = 2 - S + 2 \cdot (S - 1) \cdot \frac{p - 1}{N - 1} \tag{1}$$

allowing the selective pressure to be in the interval $S \in [1, 2]$ and for the nonlinear ranking method is can be calculated as

$$f_{vir}(p_x) = \frac{N \cdot X^{p-1}}{\sum_{i=1}^{N} X^{i-1}} \tag{2}$$

with X being computed as the root of the polynomial

$$0 = (SP - N) \cdot X^{N-1} + \sum_{i=N-2}^{0} SP \cdot X^i \tag{3}$$

and so allowing a higher selective pressure $S \in [1, N - 2]$. Once the virtual fitness values are assigned to the individuals in the sorted set, the selection is performed using a uniform random number generator in the same manner as performed in the fitness proportionate case: individuals with a higher virtual fitness have a greater slice on the roulette wheel and so are more likely to be chosen.

According to [5, 109] the rank based selection method is superior to the simple, fitness proportionate selection method. That is, when using the fitness proportionate selection method unfavorable conditions may lead to a problem that is referred to as "the scaling problem". Imagine a GA that is calculating a maximization problem with an initial population in which the fitness values range from 100 to 1100. Furthermore, let the average fitness value in this population be 550. According to the definition above, the *selective pressure* can be calculated as $S = 1100/550 = 2$. Next, imagine we are at a future stage of the GA, and the fitness values in the current population now range from 1000 to 1200 with an average fitness of 1100. It can be clearly

seen that the *selective pressure* $S = 1200/1100 = 1.09$ has become significantly lower. This effect can cause a GA to stagnate [78]. The rank based selection methods prevent this effect by conserving the selective pressure over the generations—because the virtual fitnesses remain the same, only the order of the individuals changes. One other major advantage of the rank-based selection method is that is allows for an easy implementation of *minimization problems* by assigning the virtual fitnesses anti-proportionally to the actual fitness value.

3.2.3 *Elitism*

Sometimes, good candidates can be lost when the use of the cross-over or mutation operator results in child individuals that have a lower fitness value than its parents. Of course, the GA can re-discover these lost improvements in a subsequent generation but there is no guarantee for that. To tackle this problem, a strategy called *elitism* is often used. Elitism involves copying a small number of the fittest candidates, unchanged, into the next generation. This can sometimes have a great positive impact on performance of the GA [66]. Again here, the number of individuals affected is a parameter which has to be chosen problem-specific.

3.2.4 *Cross-over Schemes*

Once two individuals are selected as the parent individuals, their "genetic" material has to be combined into a new—preferably better—child individual. There are numerous methods to recombine the genetic material of two chromosomes of which we will present the three most popular ones [48]. At this point it is worth to mention that the performance of the cross-over operator in large parts depends on the way the "solutions" to the optimization problem are coded on the chromosome. Hence, the question about the encoding should have a very high priority when considering to solve a problem using GAs. Also, it is crucial to elaborate on which cross-over method is suited best for a specific chromosome encoding. In all cases, the cross-over operator is applied to pairs of individuals which have been selected by the selection process from above.

When using the *single-point cross-over* method, the selected pair of chromosomes undergoes cross-over as follows: Let l be the chromosome length, then a random number $r \in \{1, \ldots, l\}$ is selected using a uniform random number generator. As depicted in Figure 8, the two child individuals are then formed by taking the alleles from beginning of chromosome to the cross-over point r from one parent and the rest from the second parent. The second child is formed vice versa.

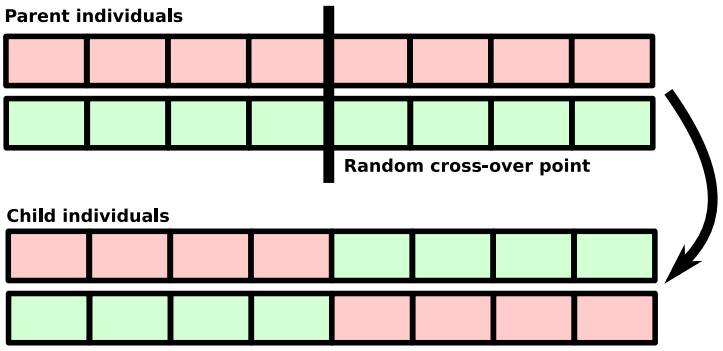


Figure 8: Illustration of the single-point cross-over method.

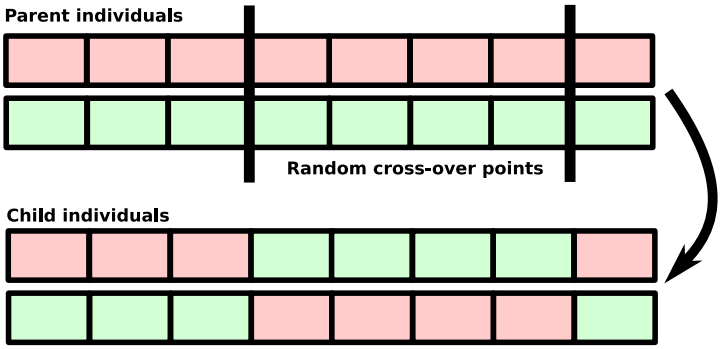


Figure 9: Illustration of the two-point cross-over method.

The *two-point cross-over* method (cf. Figure 9) works similar to the *single-point cross-over* method: two random numbers $r_1, r_2 \in \{1, \ldots, l\}$ with $r_1 \leqslant r_2$ are selected using a uniform random number generator. The first child individual is then formed by taking alleles in the range $[r_1, r_2]$ from the first parent and the rest from the second parent. The second child is again formed vice versa.

These two cross-over methods have the characteristic that chains of alleles tend to stick together: that is, alleles that were neighbors in one of the parents have a high chance to end up being neighbors on the child chromosome as well (unless they were spanning one of the random cutting points). Depending on the problem instance, such behavior may not be desirable.

The *uniform cross-over* method overcomes this problem. Here, the child individuals are formed by giving the allele of one parent to one child and the allele of the other parent to the second child while randomly choosing which allele goes to which child (cf. Figure 10).

3.2.5 *Mutation Schemes*

One of the most important factors that determines the performance of the GA is the *diversity of the population* [13, 100]. In this context, the diversity is defined as the average hamming distance between the individuals' chromosomes in the population. In the progress of a GA,

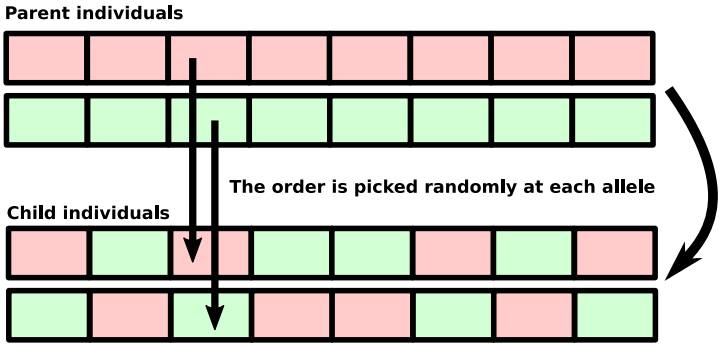


Figure 10: Illustration of the uniform cross-over method.

the diversity of the population may become too low. This in essence means that the individuals in the population have become "too similar" resulting in a low performance of the GA. More precisely, the GA can find itself in a local optimum. One way to maintain diversity in a population and prevent such local optima is to apply the mutation operator to each child individual after the cross-over took place. In essence, each bit in each chromosome is checked for possible mutation by generating a random number between zero and one and if this number is less than or equal to the given mutation probability p then the bit value is changed. In this context, the mutation probability p is recommended to be chosen very low [37, 97]: a too high mutation rate can cause the GA to behave like a random search. The best value for the mutation rate is of course problem specific, however, the authors of [37, 97] give some general guidance on how to chose the mutation rate p.

3.3 EVALUATION SETUP

3.3.1 *The Road Network Structure*

In the model that we use for evaluation, the road map is represented by a graph $G = (V, E)$ that consists of a set of vertices V that represent the intersections of the road network and a set of directed arcs $E \subset V \times V$, which describe the existing roads as directed connections between pairs of vertices. Furthermore, assume that there are z cars $c_1, \ldots, c_z$ in the road network, with car c starting its journey at time t_0^c. The route of car c is a sequence of k_c links, i. e., $R^c = (l_1^c, \ldots, l_{k_c}^c)$, where $\forall i \in \{1, \ldots, k_c\} : l_i^c \in E$ and for $i \in \{2, \ldots, k_c\}$ it holds that the i-th link starts at the vertex where the $i-1$-st link ended.

Each link $l \in E$ is associated with a function f_l; $f_l(t)$ is the travel time that it will take a vehicle entering link l at time t to traverse the link. This function depends on the traffic density and the characteristics of the respective road segment. The functions f_l are therefore, in

turn, influenced by the route choices of the cars in a non-trivial way. This is what makes the optimization problem more complex.

The time at which car c has traversed the i-th link along its route is

$$t_i^c = t_{i-1}^c + f_{l_i^c}(t_{i-1}^c), \tag{4}$$

and the car's total travel time is given by

$$T_c = \sum_{i=1}^{k_c} f_{l_i^c}(t_{i-1}^c) = t_{k_c}^c - t_0^c. \tag{5}$$

3.3.2 *Fitness Function*

In order to find the "best" routes for all cars, it is first necessary to define by which measure this decision is to be made. There are many possible choices for the fitness function. We consider two ways to formulate the fitness function here. Most heuristic optimization algorithms use the average travel time as the optimization criterion. In order to be able to compare our results to a subset of these approaches, we will perform an optimization of the route assignment problem with this fitness function. More formally, an optimization based on this fitness function leads to the minimization of the following expression:

$$\frac{\sum_{i=1}^{z} T_{c_i}}{z}. \tag{6}$$

However, this optimization criterion comes with some problems. In particular, cars with longer routes are favored during the optimization. For clarification, consider the following example: given a scenario with two cars, assume that by some change in the route assignments, the first car improves its travel time by 10 % from 50 minutes to 45 minutes. The second car, however, will then have to drive 3 minutes instead of 1 minute in the original assignment, which corresponds to an increase in travel time by 200 %. When only total or average travel times are taken into account, improved route choices for the car with the longer route (car 1) will grossly dominate any improvements or deteriorations for cars with shorter routes, even tough in relation to the length or their trip these changes may be very significant.

So, in order to show that multiple kinds of fitness functions are possible, we additionally consider a different formulation for the fitness function in our evaluation. This alternative fitness function uses the geometrical mean of the relative travel time changes. This fitness function minimizes the mean relative improvement or deterioration of all cars, when comparing their travel times in an optimized route assignment to the situation in which each car is traveling along the path with the lowest travel time at free-flow speed. For car c, we

denote this free-flow travel time by $\widehat{T}_c$. We then consider the ratios $\delta_c = T_c/\widehat{T}_c$; for a car c, δ_c is the relative deviation from the best possible travel time for car c. The fitness function is then given by the (geometric) mean of these values:

$$\sqrt[z]{\prod_{i=1}^{z} \delta_{c_i}} = \sqrt[z]{\prod_{i=1}^{z} \frac{T_{c_i}}{\widehat{T}_{c_i}}}. \tag{7}$$

Note that, strictly speaking, it is not necessary to know the free-flow travel times $\widehat{T}_{c_i}$ in order to perform the optimization: it holds that

$$\sqrt[z]{\prod_{i=1}^{z} \frac{T_{c_i}}{\widehat{T}_{c_i}}} = \frac{\sqrt[z]{\prod_{i=1}^{z} T_{c_i}}}{\sqrt[z]{\prod_{i=1}^{z} \widehat{T}_{c_i}}}, \tag{8}$$

where the denominator is independent from the current route assignment and thus constant. Minimizing (7) is therefore equivalent to minimizing

$$\sqrt[z]{\prod_{i=1}^{z} T_{c_i}}. \tag{9}$$

Nevertheless, we will report results using (7), so as to use values with a clear meaning in the above discussed sense. However, since the formula in Equation 6 is the target function that is most commonly used by other approaches, we use the formula in Equation 6 as well as the formula in Equation 7 as the fitness function for our optimization in two separate evaluations.

3.3.3 *GA Optimization Approach*

In this chapter we use a GA approach. In our case, an individual in a GA population corresponds to an assignment of route choices. That is, an individual is a vector of route choices, one for each single car in the scenario. These route choices are encoded in a bit field, which constitutes the individual's chromosome.

The first generation of individuals is generated randomly. For all the individuals, the fitness function is evaluated, then genetic operators are applied. The main intention is to preserve the beneficial properties of good individuals, while maintaining a sufficient diversity in the population in order to find yet better combinations of route choices in the search space.

3.3.4 *Limiting the Search Space*

It is, however, computationally infeasible to consider the complete search space. This space is composed of all possible combinations

of all possible routes for each single car. There is a huge number of possibilities to get from one point to the other (or even an infinite number of options when loops are allowed). Considering this search space in full is clearly not wise—especially since it can reasonably be constrained.

When designing the search space within the context of road traffic assignment, it obviously makes sense to only include plausible routes. So, in order to encode the cars' route choices for an individual into a chromosome without generating a too complex solution space, we have to think about how to model the search space efficiently. Of course—as you will see later in this thesis—multiple models of the search space are possible, each one of them fitting best into a different use case. In this case, since we assume global knowledge, we limit the number of possible route choices for each car to a fixed set of k alternative routes for each OD pair.

Those alternative routes can be found using a k shortest paths algorithm. Generally speaking, the k shortest paths problem involves finding k paths connecting a given source and destination pair in a graph. This is achieved by finding the shortest path as well as $k-1$ different paths in order of increasing costs. In order to judge which shortest paths algorithm is best suited, observe that the alternative routes have to be good choices from the drivers' perspective.

In our evaluation we have decided to use Yen's k shortest paths algorithm [115]. This algorithm finds k alternative shortest paths between two nodes in a graph while strictly avoiding loops. This is more viable from the drivers' point of view than the more generalized approach which allows cycles of repeated vertices. This variant of the k shortest path problem was proposed by David Eppstein [38]. However, driving in loops is analog (or worse) to interrupting a driver's journey in order to wait until traffic conditions become better. This clearly contradicts the typical driver's behavior and would likely not constitute a viable option.

By running Yen's algorithm on the road network for a given origin-destination pair, k alternative routes are obtained. This set of routes does not depend on the dynamics of traffic in the road network. For any given car, the set of considered routes therefore remains static throughout the optimization. For cars with different origins or different destinations, though, the respective sets of routes will differ.

For a given car, an integer value in the range between 0 and $k-1$ can be used to denote the route choice of the car. The optimization approach is based on assigning these values to the cars so that the total travel time is minimized. The route choice of a car can be encoded in $\lceil \log_2 k \rceil$ bits, so that in a scenario with z cars a chromosome consists of $z \cdot \lceil \log_2 k \rceil$ bits. In practice, one should choose k as a power of two, so that any combination of $\lceil \log_2 k \rceil$ bits is a valid entry. Given b is the bit array containing the chromosome and $\tau \in \{0, \ldots, z\}$ the index

of car c in a ordered set. Then, the current route choice for car c is represented by the bits $b_{\tau \cdot \lceil \log_2 k \rceil} \dots b_{(\tau+1) \cdot \lceil \log_2 k \rceil - 1}$.

This encoding requires special handling by the cross-over function: we need a constraint that does not permit crossing in the middle of a car's route information. Therefore we restrict the cutting points to the positions in the bit array denoted by $(\phi \cdot \lceil \log_2 k \rceil)_{\phi \in \{0,\dots,z\}}$.

3.3.5 *Cross-Over and Selection*

Applying single point or double point cross-over is not the optimal choice for the traffic assignment problem. When picking only one or two points for chromosome dissection in an evenly distributed manner, it happens that consecutive blocks of cars are very likely to stay together. More precisely, by a very small perturbation per generation it may take very long until "sub-optimal" arrangements of consecutive vehicles get split up. Imagine a very bad sequence of route choices on a specific range on the chromosome that causes a bottleneck in the road network. From the perspective of traffic modeling a small change to this sequence might relax the traffic situation. Hence, we have decided to use the bit-wise cross-over technique, adjusted as described above to exchange only consecutive blocks of $\lceil \log_2 k \rceil$ bits describing the route choice of one car.

That is, given two individuals I_i and I_j chosen for pairing, a new individual $I = [i_1 \dots i_N]$ is formed by concatenating N blocks of $N \cdot \lceil \log_2 k \rceil$ bits where each block is taken from one of the parent individuals with equal probability. This way we cause a higher degree of perturbation when evolving into a new generation. The use of elitism (taking the ρ best individuals into the next generation without modification) at the same time avoids discarding good solutions that have already been found; here, we use $\rho = 5$.

The use of the standard (fitness-proportional) roulette wheel selection limits the GA to maximization [56]. Yet, we aim to minimize the fitness value. Hence, we instead use—as described in Section 3.2.2—the individual's rank within the population, rather than its raw fitness value, to determine its probability to be chosen. In this context, we sort the individuals ascending by their fitness value and use this sorted list as the input for the rank based selection algorithm.

3.3.6 *Evaluation of the Road Network Fitness*

In order to obtain the resulting travel times of all cars in different scenarios—those represented by the individuals' chromosomes which represent different route choice configurations—,we conduct a traffic simulation for each individual. As one population is a set of many individuals, these simulations can be performed independently and so they can be easily parallelized. The framework that we

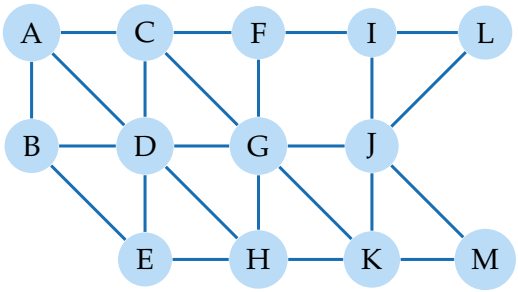


Figure 11: The structure of the road network.

developed for this evaluation has the ability to distribute those simulations among connected computation nodes, and gathers the results of each simulation run prior to evaluating the set of individuals into a subsequent generation.

But before any evaluation can be made, certain variables of the simulation have to be specified. This means that a road network has to be defined and all vehicles have to be configured with a departure time as well as with a start and destination point. In our particular case, we set the departure times in a way that they match the inflow rates in the macroscopic scenario that we are going to compare against.

In order to obtain the free-flow travel times $\widehat{T}_{c_i}$ for the fitness function (7), a separate simulation is performed for each origin-destination pair, where the specific vehicle travels along its shortest path without any congestion at all.

For the optimization, initially a random population is generated. Any individual is the evaluated by automatically setting up and running a SUMO simulation where the cars follow the routes assigned to the in the respective case. That is, the chromosome is converted into a simulation configuration, and SUMO is executed. The simulation results in a log file, from which the individual travel times can be collected, which in turn can be used to obtain the value of the fitness function. The GA then proceeds by generating a new generation.

3.4 EXPERIMENTS

In order to evaluate the effectiveness of our global optimization algorithm, we performed experiments with the road network depicted in Figure 11. This road network was derived from the optimization setting on a macroscopic model as described in [35]. The authors define four OD pairs (AL, AM, BL, BM) for evaluation. They furthermore define the link costs as the travel time between two nodes in minutes. The total demand is 1700 cars per hour, distributed over four OD pairs as follows:

$$\begin{aligned} A - L &= 600\text{cars/h} & A - M &= 400\text{cars/h} \\ B - L &= 300\text{cars/h} & B - M &= 400\text{cars/h} \end{aligned}$$

The contribution of the method proposed in this chapter is twofold. On one hand we aim to achieve a good approximation of an optimal distribution of cars to alternative routes between their origin and destination points, so that the overall system performance (based on a global fitness function) is optimal. On the other hand we also aim to maximize the convergence rate of the GA approach to make it as efficient as possible.

3.4.1 *Calibrating Parameters*

A critical parameter in our approach is the number of route alternatives available for each car, that is, the value of parameter k. In order to determine a good value for k, we ran our GA optimization with different choices. All optimizations were performed in the road network depicted in Figure 11 with the demand described in Section 3.4. We have used a population size of 100 individuals, cross-over and selection as described in Sec. 3.3.5 and a mutation probability of $p = 0.01$ per bit. Figure 12 and Figure 13 depict the progress of the the two fitness functions for the best individual over the first 20 generations for different values of k.

It can be seen that in both cases $k < 8$ keeps the search space too small: apparently good alternatives are not considered and therefore the fitness function remains at comparatively bad outcomes. More precisely, the plot indicates that the algorithm is converging slowly to a sub-optimal value. This leads to the assumption that a good distribution cannot be found when taking only $k < 8$ alternatives per car into account. Although $k > 8$ leads to similar results as $k = 8$, it takes longer until the respective values of the fitness functions are approached. One can say that the additional alternative routes give no benefit. Instead they slow down the optimization because of a significantly larger search space. We therefore use $k = 8$ in the following.

3.4.2 *Comparison to Alternative Approaches*

We compare the performance of our GA based optimization algorithm to a number of other approaches. Firstly, we take a look at two common approaches to the traffic assignment problem, i. e., approaches that can be considered the default procedure and which aim for a user equilibrium assignment and work on the macroscopic level. While it is always difficult to compare macroscopic with microscopic approaches, we include this comparison to show that the result we achieve looks "similar" to the results others have obtained with completely different methodologies.

The first approach is, according to Section 2.1, to load the network incrementally in ρ stages. That is, to assign a given fraction p_ρ (i. e., 10%, 20%, etc.) of the total demand (for each OD pair) at each time.

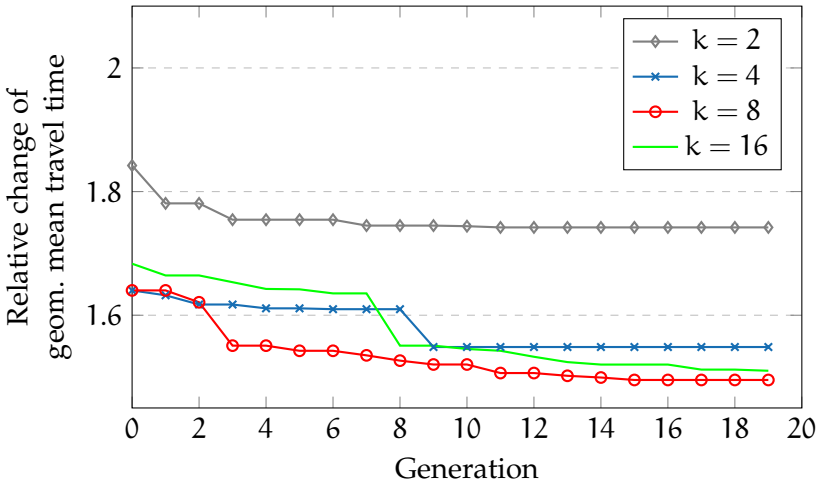


Figure 12: Progress of best individual for different values of k and the relative change of the geom. mean travel time as the fitness function.

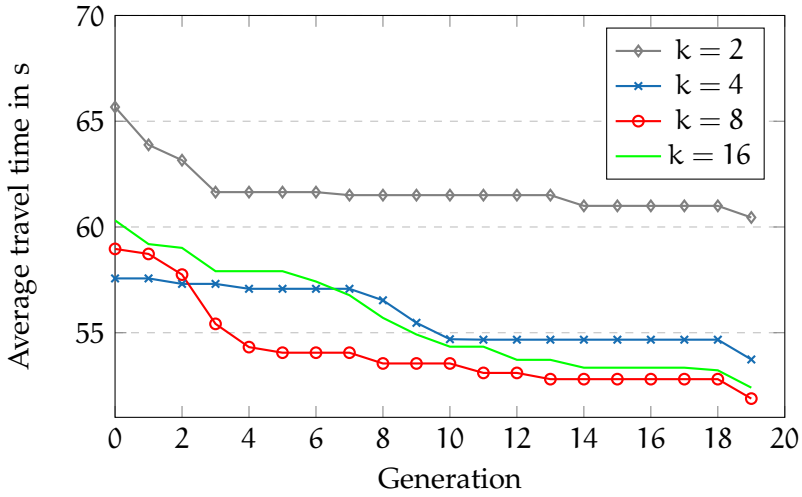


Figure 13: Progress of best individual for different values of k and the average travel time as the fitness function.

Further fractions are then assigned based on the newly computed link costs. This procedure continues until 100% of the demand is assigned. Typical values for fractions p_ρ are 0.4, 0.3, 0.2, and 0.1. The algorithm that we have used is the following one which has been adapted from [35]:

1. select an initial set of current link costs (usually the free-flow travel times); initialize flows at all links $\forall\kappa$: $V_\kappa = 0$; select a fraction p_ρ of the OD-matrix T such that $\sum_\rho p_\rho = 1$; make $\rho = 0$
2. build the set of minimum cost trees (one for each origin) using the current costs; $\rho \leftarrow \rho + 1$
3. load $T_\rho = p_\rho T$ all-or-nothing (as described in Sec. 2.6) to these trees, obtaining a set of auxiliary flows F_κ; accumulate flows on each link: $V_\kappa^\rho = V_\kappa^{\rho-1} + F_\kappa$
4. calculate a new set of current link costs based on flows V_{κ^ρ}; if not all fractions of T have been assigned, proceed to step 2.

It must be remarked that there is no guarantee that this algorithm converges to the Wardrop equilibrium, no matter how small each p_ρ is. This procedure has the drawback that once a flow has been assigned to a link, due to the accumulated nature (see step 3), it is never removed. Thus, in case an arbitrarily low over-capacity is assigned to a link, it prevents the convergence to the optimum solution. However, it is very easy to implement.

The other approach is to start from some initial values for the link costs and find the minimum cost routes. Trips are then assigned to these routes. New costs are computed and this cycle is repeated until there is no significant change in link or route volumes. For instance, in the method of successive averages, the flow at the ρ-th iteration is calculated as a linear combination of the flow on the previous iteration and an auxiliary flow resulting from an all-or-nothing assignment in the ρ-th iteration.

This can be formalized as the following procedure (again, adapted from [35]):

1. select an initial set of current link costs (usually the free-flow travel times); initialize flows at all links κ and make $\rho = 0$.
2. build the set of minimum cost trees (one for each origin) using the current costs; $\rho \leftarrow \rho + 1$.
3. load the whole of the matrix T all-or-nothing to these trees obtaining a set of auxiliary flows F_κ.
4. calculate the current flows as: $V_\kappa^\rho = (1-\phi)V_\kappa^{\rho-1} + \phi F_\kappa$, with $0 \leqslant \phi \leqslant 1$.

5. calculate a new set of current link costs based on V_k^ρ; if no V_k^ρ have changed significantly in two consecutive iterations, stop; otherwise proceed to step 2 (or, alternatively, use a maximum number of iteration).

The last step of the successive averages method admits several ways to fix the value of ϕ. A useful one is to make $\phi = 1/\rho$. There is a proof that this produces solutions convergent to the Wardrop's equilibrium; this, however, may be very inefficient.

Finally, we also compare our GA to the microscopic dynamic user equilibrium (DUE) [45] which is also obtained using an iterative approach (cf. Chapter 2) and is implemented directly in the SUMO simulator.

3.5 EVALUATION RESULTS

Figures 14 and Figure 15 illustrate the overall progress of the GA for $k = 8$ alternative routes per OD pair in our scenario for the two fitness functions described above. In both cases, the x axis represents the progress in generations and the y axis shows the best individual's fitness value. The graphs contain five plots, which depict the fitness functions of the best individual, the .25 percentile, the median, the .75 percentile and the worst individual, respectively.

As we are starting with a random initial population, we can observe that the fitness function values of the individuals in generation 0 are widely spread. Over the generations our algorithm decreases the fitness function. The plots indicate that not only the fitness function (which, we remind, is to be minimized) of the best individual decreases, but also that the homogeneity increases. The fluctuating spikes of the worst individual can be explained with the random mutations occasionally creating a few very bad individuals in each generation.

We have also investigated how the population's diversity develops in the course of the optimization. Figure 16 and Figure 17 depict the diversity of the population measured by the average Hamming distance between each pair of individuals in the population, again for both fitness functions. The Hamming distance, in this context, is defined as the number of set bits when the chromosomes of two individuals are `xor`'ed. More formally, the Hamming distance between two chromosomes x and y is defined as $d(x, y) = |\{1 \leqslant i \leqslant n \mid x_i \neq y_i\}|$, where x_i refers to the i-th bit of x.

We have a total of 1700 cars in each simulation encoded on each individual. With $k = 8$ route alternatives (which can be coded on three bits) this results in a genotype with a length of 5100 bits. The random population (generation 0) has an average Hamming distance of around 2500. This indicates that we start off with a fairly random population of high diversity. During the progress of generations, as both

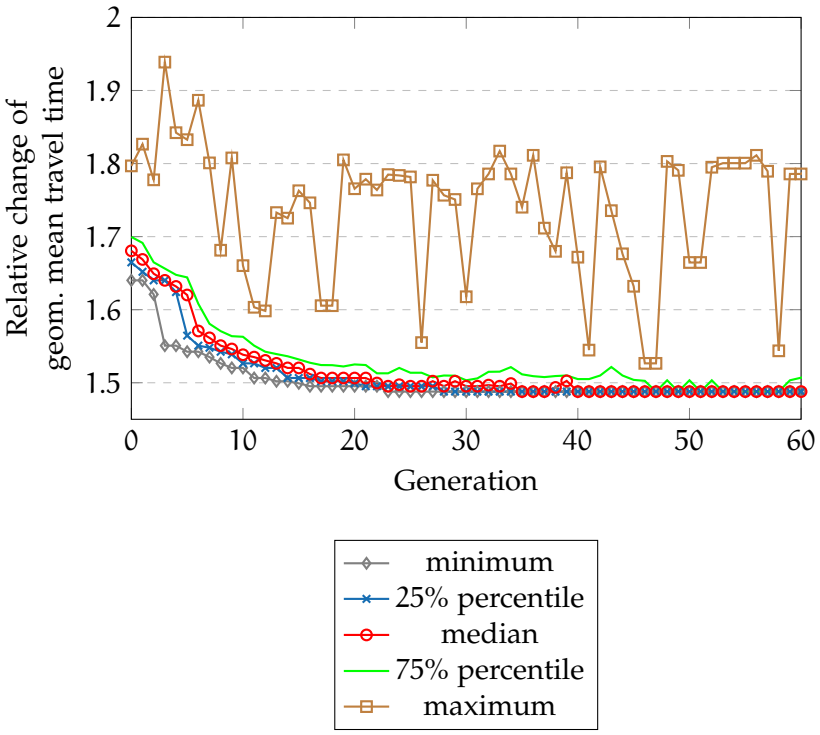


Figure 14: Relative change of geom. mean travel time over generations in GA population.

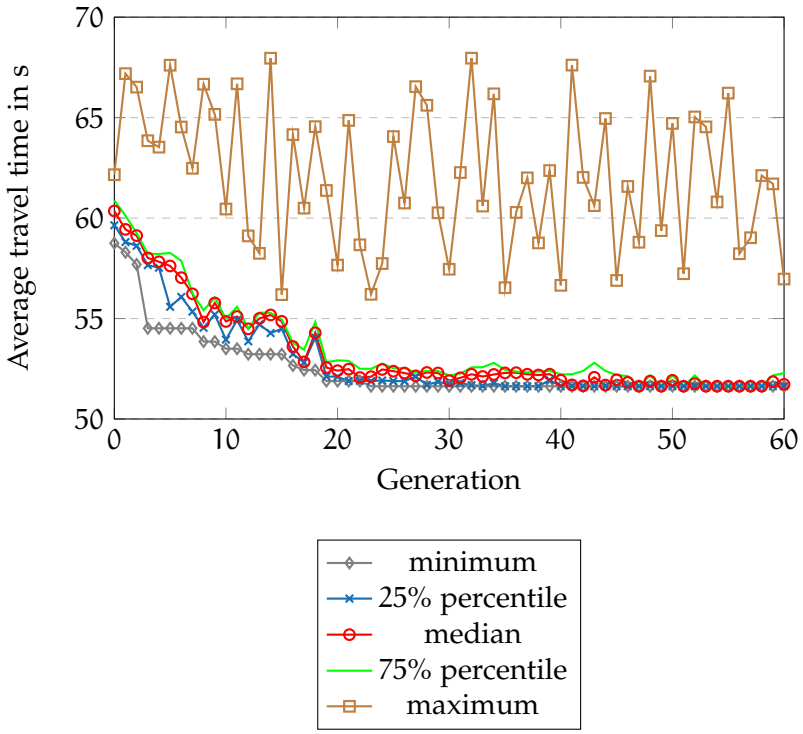


Figure 15: Average travel time over generations in GA population.

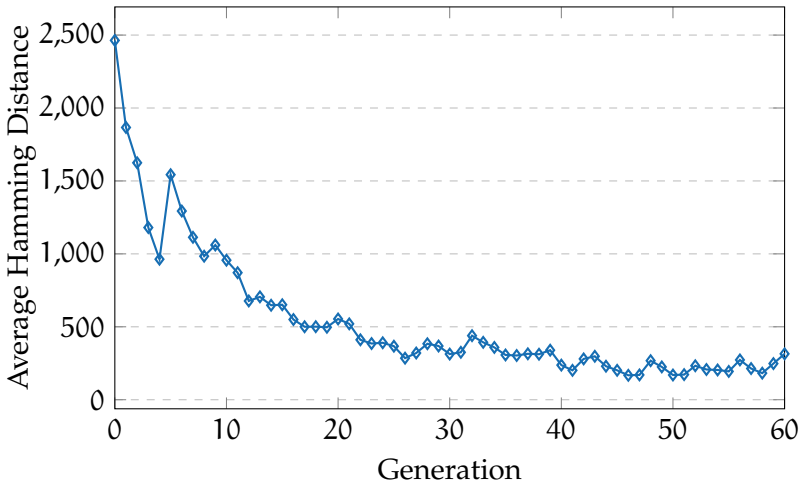


Figure 16: Average Hamming distance over the generations with the the relative change of the geom. mean as the fitness function..

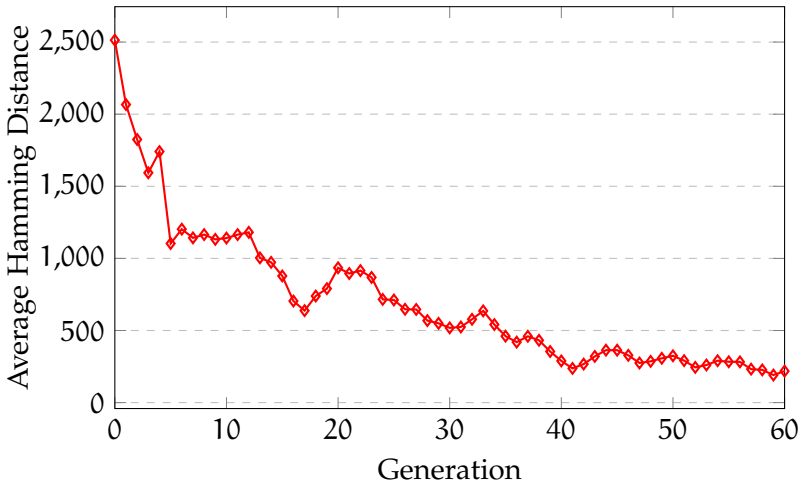


Figure 17: Average Hamming distance over the generations with the average travel time as the fitness function.

plots show, the diversity falls indicating an increasing homogeneity of the population itself. However, it can be seen that the diversity has not dropped too low before the fitness function converges (which occurs roughly after the 20th generation).

After all, in our evaluation we have not been able to show a difference in the algorithm's performance for the two different fitness functions.

Next, we discuss the comparison of the performance of our GA to other traffic optimization approaches proposed in literature (see Sec. 3.4.2). Most of these approaches use the average travel time in the network as an optimization metric, i. e., they optimize according to (6). To recall, we have performed two evaluations; one for the average travel time and one for the relative change of the geometric mean of all travel times. For better comparability, we—in the case of the second target function—therefore also evaluate the best individual found by our approach again in terms of the average travel time as well. This involves taking best solution achieved by optimizing towards Equation 7 and using the average travel time as described in Equation 6 for the comparison to the other approaches.

As depicted in Table 1, we have performed a per-OD-pair analysis of the average travel times and compare them to those obtained when optimizing with alternative approaches. The "All-or-Nothing" (AON) column indicates the travel times along the shortest path under no congestion. The approaches described in Sec. 3.4.2 are abbreviated by "Incremental" (Inc.) and "Successive" (Suc.). The column titled "one-step SUMO" (One) shows the values for a microscopic evaluation of our scenario without any optimization, i. e., cars use the route that is computed by SUMO, which is based on Dijkstra's shortest path algorithm [33] using the free-flow edge traversal time as the edge weights. The "DUE" column shows the results by performing an iterative loading approach in our microscopic traffic model (achieved by SUMO's toolset DUA-Router) which aims towards a dynamic user equilibrium. Finally, the "Global GA" column distinguishes between two cases: the usage of the average travel time (Avg) and the usage of the relative change of the geometric mean of all travel times (Rel) as the fitness function.

We can clearly observe that the GA is not necessarily improving individual OD pairs. For example, and this is the case for both fitness functions, the GA causes an increase in the average travel time for the OD-pairs A–L and A–M in comparison to the DUE. This is not surprising as the optimization goal is not to generate ideal results for particular cars, or for individual OD-pairs, but for a global optimization criterion. As the "overall" row shows, this goal is clearly achieved by our approach for both target functions.

Table 1: Comparison of different optimization approaches using the average travel time as the fitness function.

		Other approaches					Global GA	
OD	Trips	AoN	Inc.	Suc.	One	DUE	Rel	Avg
A–L	600	28.00s	67.80s	70.66s	74.79s	55.17s	57.00s	56.25s
A–M	400	26.00s	66.20s	64.82s	70.53s	49.17s	53.50s	54.13s
B–L	300	32.00s	69.40s	68.96s	51.95s	64.74s	55.15s	54.96s
B–M	400	23.00s	63.00s	62.22s	44.58s	52.79s	43.80s	43.09s
				Overall Results				
	1700	27.06s	66.58s	67.00s	62.65s	54.89s	52.74s	52.42s

3.6 CHAPTER SUMMARY

This chapter has demonstrated that GAs are a feasible approach to the route assignment problem in road networks. We have shown a way to approximate the optimal assignment of routes to a number of vehicles in a given scenario. This optimization is performed using an on-top view on a microscopic traffic model. Compared to numerous well-known approaches to traffic assignment, we have shown that our approach yields better results. This again shows us that existing methods do not make full use of the optimization potential. At the same time this methodology forms the basis for the remainder of this thesis. More precisely it allows us to assess the overall optimization potential of arbitrary road networks and gives us a "best case" measure for the comparison of other approaches—namely those that we present in the subsequent chapters of this thesis.

4

LARGE-SCALE GRAPH-BASED TRAFFIC SIMULATIONS

4.1 CHAPTER OVERVIEW

Whenever it is necessary to perform a large number of simulations, the traffic simulation—depending on the detail level required—regularly becomes a bottleneck. Using regular microscopic traffic simulators such as SUMO, the computation time for the microscopic evaluation of one single mid-size scenario is usually already in the order of minutes. This represents a major problem for our GA approach (cf. Chapter 3): in order to evaluate the fitness of each individual, it is necessary to "observe" the traffic conditions in the network in order to determine the average travel time, the geometric mean of all travel times, or any other target value that is used as the fitness function of the cars. This observation is conducted under the premise that the cars follow the routes as described in this individual's chromosome. In our proposed approach—as we are trying to optimize individual route choices—this is achieved by using a microscopic traffic simulation. A macroscopic simulation would, due to the aggregated nature, not be sufficient for accurately evaluating the impact of individual route choices of specific cars.

On this basis, it becomes computationally expensive to evaluate all possible route assignments on all chromosomes over multiple generations in a feasible time. The evaluation setup that we have described in Section 3.3 (this includes all required simulations for 20 iterations of our GA) alone took around eight hours to terminate on a high-end computer with 64 GB of RAM, an Intel Xeon E5-2630 processor and all files stored in a ram disk. Evaluations in larger, more complex traffic scenarios would barely be possible in a reasonable time frame. In fact, this problem becomes critical in the remaining chapters of this thesis, when we will be required to conduct an evaluation using a large number of traffic simulations which is in the order of hundreds of thousands.

In the remainder of this chapter we propose an adaptation of Garwon's queuing model [44], respectively the MATSim [25] model. Both models allow for a fast microscopic queue-based traffic simulation which trades precision for efficiency and speed. This is mainly achieved by dropping the simulation of the actual movement of the cars—to recall, this is achieved with a complex car following model that models the interdependencies between multiple cars—in favor of a fast and light weight queuing model. To be precise, the traversal

of a link in these models is described by only two events: the entrance and the exit of a queue (which constitutes the edge that the car is traversing). The two models, however, lack a complex intersection model which has a negative effect on the quality of the simulation result. While this effect probably levels out in long term analyses, a short time traffic prediction might become problematic.

Our contribution suggests a new, adapted model which allows for a fast, queue-based traffic simulation and, at the same time, is superior to present approaches such as MATSim in terms of precision. This is achieved through the support of junctions with any arbitrary junction logic. This contribution constitutes one key aspect that will allow the proposed GA approach to be executed quicker and so form the foundation for online traffic optimization.

4.2 GAWRON'S QUEUE MODEL

Gawron introduced a queue model, which describes the traffic dynamics on a graph-based traffic simulation [44]. Let $G = (V, E)$ be the road network graph, which consists of a set of vertices V that represent the intersections of the road network, and a set of directed arcs $E \subset V \times V$, which describe the existing roads as directed connections between pairs of vertices. Each link $e_i \in E$ is associated with parameters which describe the characteristics of the corresponding road segment. According to Gawron, these characteristics are: a) the free-flow travel time; b) the storage constraint; and c) the flow capacity constraint of a road segment. The free-flow velocity $v_{f,i}$ of link e_i is the speed at which the non-congested road segment can be traversed. Given the physical length l_i of the road segment, this implies the free-flow travel time $t_i = \frac{l_i}{v_{f,i}}$, meaning the minimum time for traversing the link. The storage constraint describes the maximum number of cars a link can hold at any given time. Let δ be the space a single vehicle on average occupies in a jam, then $N_i = \frac{l_i}{\delta}$ denotes the maximum number of cars the link can hold at any given time. The flow capacity constraint M_i specifies the number of cars that can leave a road segment in a given time interval τ.

In Gawron's queue model, cars are moved from one road segment to another along their desired route until they reach their destination. The model does *not* keep track of the current position of the cars on the link; this is the property which simplifies the necessary computations to an extent that allows for sufficiently fast evaluation of the fitness function in the context of our proposed GA scheme. Rather, the movement is described only in terms of link entry and link exit times and is based on the following three rules:

- A vehicle entering a link i at time T is added to the link's queue. The earliest time it can be dequeued and moved to the subsequent link is $T + t_i$.

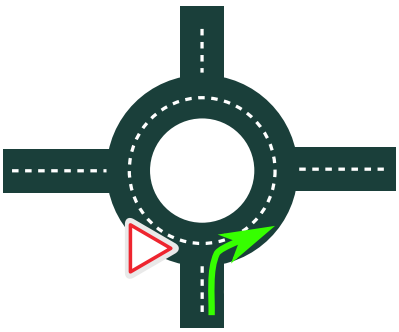


Figure 18: Roundabout.

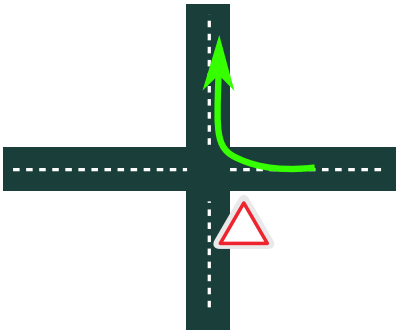


Figure 19: Right-before-left.

- If the subsequent link e_{i+1} is full (i. e., already holds N_{i+1} cars), vehicles cannot be moved across the intersection.
- Vehicles can only be dequeued and put on the subsequent link if the flow capacity constraint has not been met. This is, the number of cars which left e_i within the current time frame of length τ has not yet exceeded the link's flow capacity constraint M_i.

In Gawron's queue model, links are selected and processed sequentially in an arbitrary but fixed order. This can lead to unrealistic behavior under congested traffic conditions. For example, a small side road can block a major street when it is processed before the major road after the space on the subsequent link has opened up. This problem has been mitigated in MATSim [76] through the use of an intersection-based queuing model. The key idea is to move from a link oriented queue model, as proposed by Gawron, to an intersection based queuing model. When an intersection determines that there is space on the subsequent links, it randomly picks one of the incoming links—here, links with a higher density have a higher probability of being chosen—and processes all waiting cars that can be processed without violating any of the three rules described above. This concept can be seen as a load balancing technique with a dequeuing method that is based proportionally on the traffic load on the links.

Load-balancing junctions are not sufficient from our point of view. Therefore, our goal is to reuse the general idea of this model and adapt the queuing discipline inside the intersection in a way that it supports numerous junction logics. Junction logics that can be found in common road networks, such as right of way, all-way stop and right before left.

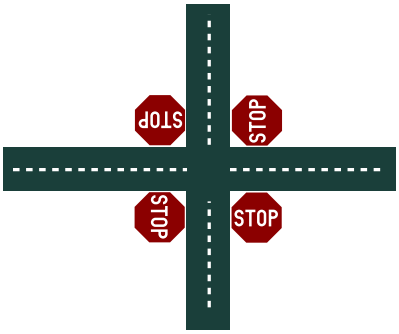


Figure 20: All way stop.

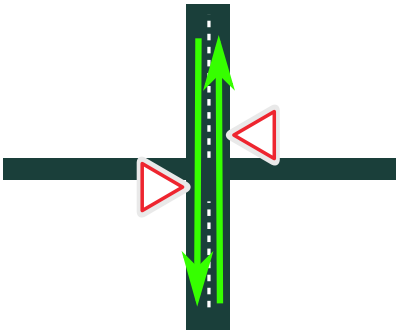


Figure 21: Right-of-way.

4.3 ARBITRARY JUNCTION LOGIC

4.3.1 *Junction Types*

While both Gawron's queue model and the changes to the intersection logic that were applied by MATSim approximate the traffic dynamics in terms of the fundamental diagram [11], they just approximate the behavior of a macroscopic simulation. More precisely, the junction logic does not reflect the complex dynamics observed in real life when taking a closer look at the situation—a microscopic look. A junction, which coordinates traffic only based on the demands of the incoming links does not find itself in both reality and state of the art microscopic traffic simulators. Quite the opposite is the case: there are multiple junction types, each leading to completely different traffic patterns. Four of the most popular junction types and their behaviors are:

- In **right-before-left junctions** (cf. Figure 19) priorities are not based upon the density of the incoming lanes, but simply on the spatial relations of the links to each other. While two equal flows would be treated completely fairly in the MATSim fair intersection model, in reality one flow might get completely blocked in a right-before-left junction situation.
- **Right-of-way** junctions (cf. Figure 21) are very similar to this case: certain roads just have a hard coded priority over the others.
- **Allway-stop junctions** (cf. Figure 20) are junctions which basically have a stop sign at all incoming links and serve the queuing vehicles in a round robin manner. In this context, the fair intersection model of MATSim would prioritize the flows depending on their actual load while the correct approach would be to give all links an equal priority.

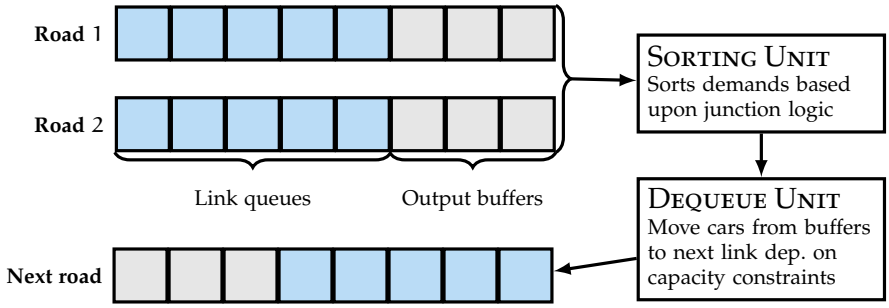


Figure 22: Logic based junction controller for Gawron's queue model.

- **Roundabouts** (cf. Figure 18) are a special type of junctions that are modeled by a concatenation of subsequent right-of-way junctions.

In order to approach the behavior observed in reality as well as reflected by common microscopic traffic simulators it is therefore necessary to introduce a more sophisticated junction logic for Gawron's queue model.

4.3.2 *Multiple Queue Approach*

In this section we introduce an adaptation of Gawron's queue model to support arbitrary junction controller logic. In order to stay within the bounds of the three boundary conditions as described in section 4.2, we propose the following adaptations. We rely on the original link based queue model, where each link consists of a queue (which holds all cars that are currently transiting the link) and an output buffer (which contains all cars that already traversed the link and wait at the end of the link to be placed on their desired next link). The free flow travel time constraint is met, because cars traveling along link i remain in the our approach for exactly t_i time units in the main link queue. The "Dequeue Unit" ensures that conditions two and three are met. This unit is continuously monitoring the subsequent links of the intersection. If there is space on the subsequent links, it dequeues the number of cars that fits on the subsequent links from the sorted demand list which is maintained by the "Sorting Unit". This "Sorting Unit" allows us to define an arbitrary order, in which cars may traverse the intersection. Here, to point out, lies the key difference to the strategy implemented by MATSim.

Specifically the size constraint takes both the queue and the buffer size of a link into account as the queue buffer holds all cars that are traversing the link and the output buffer holds all cars waiting at the end of a lane. Implicitly, the "Dequeue Unit" makes sure that neither the flow capacity constraint nor the space constraint are violated. A simplified flow chart of this model is depicted in Figure 22.

The junction logic itself can now be implemented easily in the "Sorting Unit" without taking care of breaking one of the three constraints formulated by Gawron. The junction logic, exemplarily only for two of the four major junction types described in Section 4.3.1, can then be implemented programmatically using the following algorithms in pseudo code. The other junction logics would be implemented analogously.

```
   input  : The subsequent link: NextLink
   output : The car that may pass the intersection
 1 S ⟵ AllQueuedCars();
 2 if IsFull(NextLink) then
 3 |   WaitUntilFree(NextLink);
 4 end
 5 foreach car in S do
 6 |   if IsEmpty(GetLaneRightOfLane(car.lane)) then
 7 |   |   return car;
 8 |   end
 9 end
10 return TopElement(S);
```

Algorithm 1 : Junction logic for the right-before-left junction.

```
   input  : The subsequent link: NextLink
   input  : Set I of all incoming links
   output : The car that may pass the intersection
 1 C ⟵ CircularIterator(I);
 2 if AllEmptyIn(I) then
 3 |   return null;
 4 end
 5 if IsFull(NextLink) then
 6 |   WaitUntilFree(NextLink);
 7 end
 8 while IsEmpty(GetLane(C)) do
 9 |   C ⟵ C.next;
10 end
11 return GetLane(C);
```

Algorithm 2 : Junction logic for the allway-stop junction.

Please note, that the exemplary algorithms presented in pseudo code here can be implemented very efficiently and constitute no significant overhead compared to the weighted-randomized selection as implemented in MATSim.

4.3.3 *Drivers' Behavior: Car Following Models*

A correct junction logic alone is not sufficient to result in a movement pattern that is similar to what we would observe in a state of the art microscopic traffic simulation. The dequeuing logic only defines the order in which the cars are passing the intersection. At fully crowded conditions—that is, when the subsequent link is full and the cars get queued up in the output buffers—this may suffice. In less dense situations, where cars may pass the intersection without waiting, this is not always the case. Thus the influences of the cars among each other (as for example in a stop and go situation when traffic queues up at the end of a lane due to conflicts with higher prioritized crossing flows) must be modeled separately. Here, several aspects have to be taken into account that we discuss in more detail now.

In order to decide whether a car z is allowed to pass the intersection or is required to let a higher priority vehicle pass, it is mandatory to know the exact time when a car is expected to arrive at the intersection point if it was allowed to pass (that means, without the deceleration phase at the end of a link when a stop is required). In order to achieve this, every junction controller maintains a sorted list of *pass times*, which are the time stamps indicating when all cars are expected to pass the intersection from any of the incoming links. Along with each such pass time, the identifier of the link that the car is entering the intersection from is saved. Furthermore, it is required to define a few driver model parameters similar to how they are used in common car following models such as the Kraus model [64], the Intelligent Driver Model (IDM) [99] or a Todoslev's action point based model by Peter Wagner [69]. In this context τ denotes the driver reaction time, l the vehicle length including a constant safety gap at zero speed, a the acceleration in $\frac{m}{s^2}$, d the deceleration in $\frac{m}{s^2}$, p_c the maximum speed and s_c the total length in meters of link c. Furthermore, $\delta \in \{0, 1\}$ describes if the vehicle entered its current link at free flow speed ($\delta = 0$) or whether it started accelerating from a prior stop ($\delta = 1$). Further let `maypass(i, lane, time)` be a function telling whether the flow coming from *lane* is prioritized at intersection *i* at any given time. In case of a allway-stop or right-before-left junction this function can be trivially implemented by taking a look into the sorted *pass times*. Also, we define $D(c) = \frac{p_c}{a} \cdot s$ as the time needed to accelerate from zero speed to the maximum speed of link c and $S(c) = \frac{1}{2} \cdot a \cdot D(c)^2$ as the needed distance for this acceleration. For now, we assume that the link length is greater than the distance needed for a full acceleration. Now we can define

$$T_{\text{estimate}}(c, z, i) = \begin{cases} \frac{s_c}{p_c} & \text{if } \delta \equiv 0 \\ D(c) + \frac{s_c - D(c)}{p_c} & \text{if } \delta \equiv 1 \end{cases} \tag{10}$$

as the estimation of the fastest possible time to arrive at the end of the lane at free-flow. Now, it is necessary to differentiate between two cases.

The first case includes arriving at the intersection point without being blocked by vehicles in front: this case is applicable when all cars, which are on the same lane and ahead of the observed car, have passed the intersection prior to the arrival at the intersection point. More formally, let Γ_i be the list of all *pass times* in intersection i, and $\psi_{c,i} \subset \Gamma_i$ a subset of that list only containing the *pass times* of the current link c. The car arrives at the end of the link without being influenced by other cars when the subset $A = \{x \mid x \in \psi_{c,i} \wedge \text{now} \leqslant \text{time}(x) \leqslant T(c, z, i)\}$ contains exactly as many elements as there are cars on the current link before en-queuing the current car to the link queue: $|A| = \text{occupancy}(i)$. This check is valid, because this process is done once prior to en-queuing a car on the link which guarantees that there are no cars on the link which are not in front of the current car.

Now, if `maypass(i, c,` $T_{\text{estimate}}(c, z, i)$`)` returns true, the car's exit time matches this estimator and the dequeue time both in our queue model and in the *pass list* of the intersection is known. As a matter of fact, the car can be enqueued in the link queue. Congestion on the subsequent link and thus a delay of this estimated exit time will be handled in the dequeue phase (by the dequeue logic) while ensuring to stay within the capacity bounds.

However, if `maypass(i, c,` $T_{\text{estimate}}(c, z, i)$`)` returns false the car will in any case have to stop at the end of the intersection. In this case, not only a possible acceleration phase has to be taken into account but also the deceleration phase at the end of the link. Define $\tilde{D}(c) = \frac{p_c}{d}$ as the time needed to decelerate from from the maximum speed of link c to zero and $\tilde{S}(c) = \frac{1}{2} \cdot d \cdot \tilde{D}(c)^2$ as the needed distance for this deceleration. Further define

$$T_{\text{no\_influence}}(c, z, i) = \begin{cases} \tilde{D}(c) + \frac{s_c - \tilde{D}(c)}{p_c} & \text{if } \delta \equiv 0 \\ \tilde{D}(c) + D(c) + \frac{s_c - \tilde{D}(c) - D(c)}{p_c} & \text{if } \delta \equiv 1 \end{cases} \tag{11}$$

as the time the car needs to reach the intersection point including a possible acceleration phase at the beginning and come to a full stop. This time does not reflect the actual passing time of the junction. In order to get the actual passing time, it is mandatory to estimate the influences of other cars competing at the junction point. In order to quantify the time delay at the intersection point we in a first step have to determine how many times the flow priorities will change before a car can pass. This is mandatory, because each time the priority is changed to another flow, one flow has come to a stop (which again will delay all subsequent cars on that lane) and the other, now priori-

tized, flow has to accelerate. These accelerations come along with the drivers' reaction time τ. Let

$$|\bigcup_{x_i \in \Gamma_i} \Upsilon_{x_i}| = \sum |\{\text{lane}(x_i) \neq \text{lane}(x_{i-1}) \wedge \text{now} \leqslant \text{time}(x_i) \leqslant T_{\text{noinfl}}(c, z, i)\}| \tag{12}$$

denote the number of flow transitions until the particular car reaches the intersection point and Φ with $|\Phi| = |\bigcup_{x_i \in \Gamma_i} \Upsilon_{x_i}|$ a set containing the transition phases; in this context $\rho_i \in \Phi$ contains the i-th block of cars which pass the intersection between two flow transitions. We now have to take the following delays into account:

1. Each car that wants to pass the intersection and was required to stop at the intersection point has a reaction time of τ.
2. Each car that wants to pass the intersection and was required to stop at the intersection point is delaying the flows for the duration of its acceleration phase.
3. The last car of each block additionally delays the flows with its deceleration phase at the end of the lane.

More formally, this accumulated delay can be written as following. Let $\delta_{\text{multiple}}(x) \in \{0, 1\}$ be the *Kronecker Delta* function, returning 0 when at most one car is in the set x and 1 otherwise. Now define

$$T_{\text{stop\_delay}}(c, z, i) = \underbrace{[\tau + D(c)] \cdot |A|}_{\substack{\text{reaction and acceleration time} \\ \text{of all queued vehicles}}} + \underbrace{|\{\rho_i \in \Phi | \delta_{\text{multiple}}(\rho_i) = 1\}| \cdot \tilde{D}(c)}_{\substack{\text{deceleration time of last} \\ \text{vehicle in each passing block}}} \tag{13}$$

as the additional delay due to stop and go at the intersection. At this point the correct passing time for the currently enqueued car is $T_{\text{pass}}(c, z, i) = T_{\text{no\_influence}}(c, z, i) + T_{\text{stop\_delay}}(c, z, i)$. The car can be enqueued into the link queue with this passing time. Congestion on the subsequent link and thus a delay of this estimated exit time will again be handled in the dequeue phase (by the dequeue logic) to stay within the capacity bounds. The second case, namely arriving at the intersection point and being blocked by vehicles in front, can be handled analogous to the case when a car is not blocked by vehicles in front of it. When adjusting the initial $T_{\text{estimate}}(c, z, i)$ to return the time, until all prior cars from the same lane have passed the intersection (which can be extracted from the *pass list* inside the junction controller) the above model remains valid.

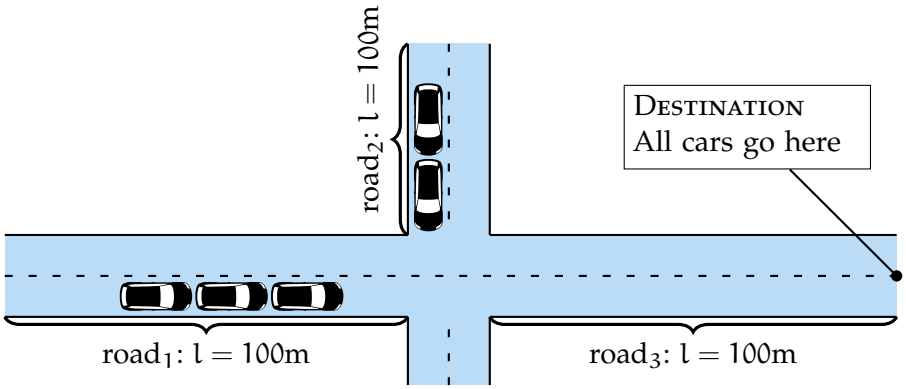


Figure 23: Very simple road network for our comparison.

4.4 EVALUATION RESULTS

4.4.1 *Comparison with Current Traffic Simulators*

This section shows the differences between the Gawron queue model as implemented in the MATSim fair intersection model, our proposed simulation approach and the microscopic simulator SUMO. The evaluation will be done with regard to the overall traffic pattern and with a detailed view at the traffic dynamics at intersection points.

We start with a scenario with two single lane roads merging into a third single lane road at an intersection. Figure 23 contains a sketch of the used network. All roads are configured to have a length of $\text{len}(\text{road}_{1,2,3}) = 200\text{m}$ and a free flow travel speed of $v_{1,2,3} = 10\frac{\text{m}}{\text{s}}$. The roads intersect in the middle with a right-before-left junction logic. This means, at free flow the cars reach after $T_{1,2} = \frac{\text{len}(\text{road}_{1,2})}{v_{1,2}} = 20$ the intersection point. The scenario has two possible OD pairs: $(\text{road}_1, \text{road}_3)$ and $(\text{road}_2, \text{road}_3)$. Both flows are configured with a constant inflow rate of one car per time unit for a total of 100 cars.

Overall Traffic Pattern

Figure 24 shows the trajectories for the queuing model as implemented in MATSim for the cars going from road_2 to road_3. In this context the x axis corresponds to the simulation time and the y axis depicts the cars' absolute position on the road. As expected, both flows get assigned with the same priority due to the same demand. This can be seen by looking at the equally distant gaps after the intersection, which are twice as large as the gaps before the intersection. This indicates a perfect zipper merge at the intersection, which is clearly not what was intended. The trajectory plots for SUMO (cf. Figure 25) and for our proposed approach (cf. Figure 26) however look what would be expected in this case. The slight difference be-

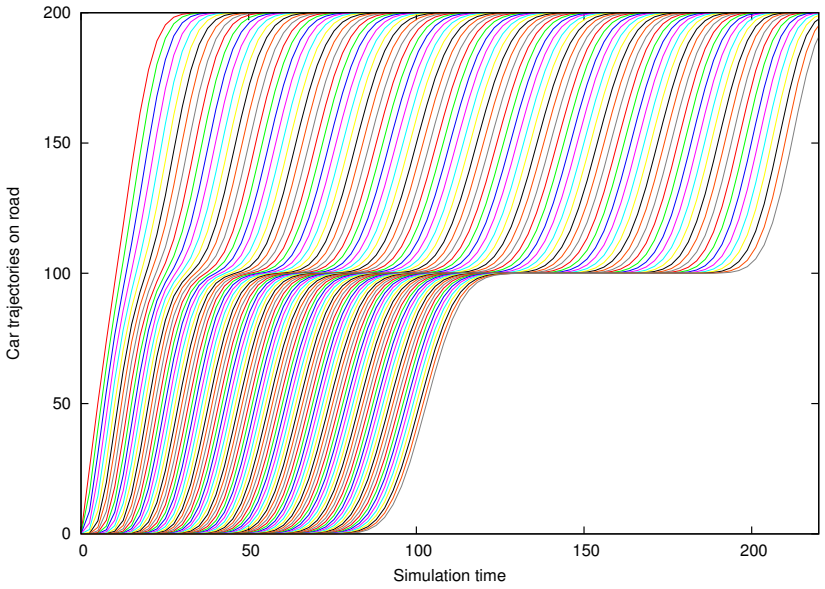


Figure 24: Trajectories for cars traveling from $road_2$ to $road_3$ in MATSim's fair queue model.

tween these two plots can be explained due to random factors in the drivers behavior modeled in SUMO (as dawdling). However the overall observation is clear: in the beginning, $road_1$ (which is coming from the right) gets complete priority due to the right-before-left logic. After the whole demand of 100 cars has been served, the lower priority lane may pass. This clearly is visible by the large waiting time that is observable in the trajectory plot.

In a large road network with many intersections of different types it can be safely assumed, that the results achieved from MATSim are not comparable to the output of full-featured microscopic simulators like SUMO. Such differences propagate through the network and cause the simulation result to be arbitrary far away from what one would expect. The biggest problem in this context are locked flows due to a specific junction logic which are not modeled in the fair junction model of MATSim.

Influences of Cars Among Each Other at Intersection Points

In this section, we will focus on the traffic dynamics, which usually take place at the intersection points due to queues which slowly build up. In this case, special treatment (like stop and go and reaction times) is required as per section 4.3.3. We start with the same network as depicted in Figure 23 and described in Section 4.4.1 but with a slight change so that links all have a length of $\text{len}(\text{road}_{1,2,3}) = 495.25\text{m}$ This time we configure a total of 10 cars of which 5 cars are assigned to each of the two OD pairs. The two flows both start at $t = 0\text{s}$ and the inflow rate is set to be $r = \frac{1}{s}$. The maximum speed on the links is

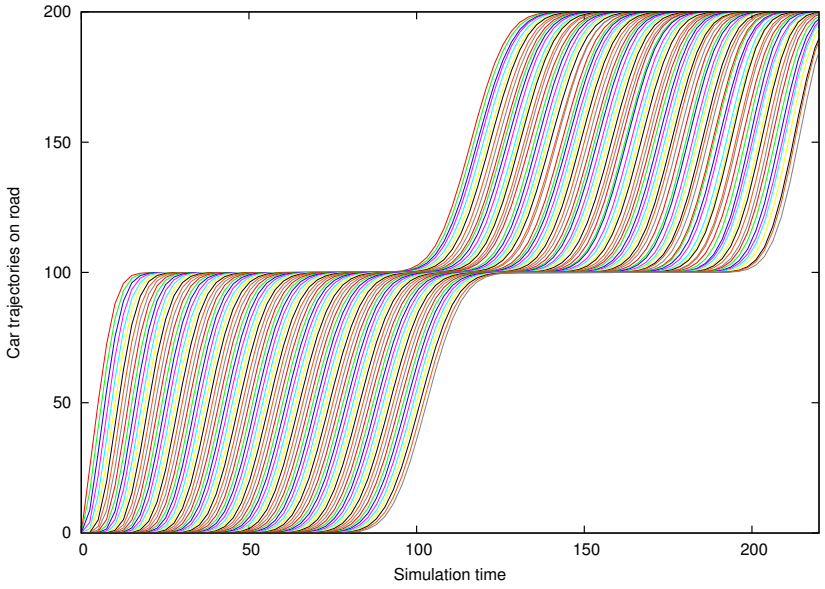


Figure 25: Trajectories for cars traveling from $road_2$ to $road_3$ in SUMO.

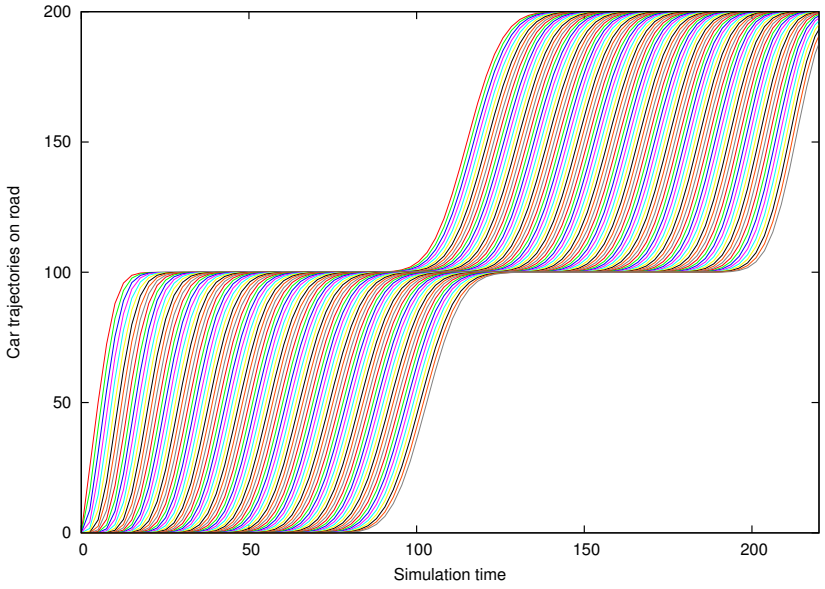


Figure 26: Trajectories for cars traveling from $road_2$ to $road_3$ in our proposed approach.

Table 2: Arrival times in SUMO simulator.

Car	Arrival time
0	68.00
1	69.00
2	71.00
3	72.00
4	75.00
5	75.00
6	78.00
7	78.00
8	81.00
9	81.00

Table 3: Arrival times in proposed simulator.

Car	Arrival time
0	68.00
1	69.00
2	71.00
3	72.00
4	74.00
5	75.00
6	77.00
7	78.00
8	80.00
9	81.00

$v = \frac{15m}{s}$ and the cars both have a acceleration and deceleration of $a = \frac{15m}{s^2}$. Further, the junction logic has been changed to the allway-stop logic. Tables 2 and 3 show the cars' total travel times both for SUMO and our reference implementation of the proposed approach. Due to a bug in the SUMO implementation we had to set the acceleration and deceleration to $a = \infty$ in this evaluation (meaning cars depart immediately).

We can see, that the arrival times are almost equal. Specifically the first and last cars have the correct arrival times. Car 0 thus needs $t_0 = 2 \cdot \frac{495.25m}{15\frac{m}{s}} + 2 \cdot \tau \approx 68s$ with $\tau = 1$ being the driver's reaction time. In this context, τ cannot be assigned with values < 1 due to the missing sub-second simulation feature in the SUMO version that was used to conduct these experiments. Also, the last car has its correct arrival time. At this allway-stop junction each car has to stop. The priority is assigned in a round robin matter to the intersecting links. Car 9, e. g., would usually arrive the the intersection after $t = 33s$. Due do the driver dynamics described above, this time is delayed. Our estimation matches the behavior as it is implemented in SUMO. The differences, as it is the case for car 4 (74s compared to 75s), have been inspected and are a result of more complex non-linear driver behavior model in SUMO. Driver imperfection (*Sigma*) and *dawdling* are not implemented in our model and can slightly increase the travel times observed in SUMO. Figure 27 is a visualization of the noticeable gaps seen in the tables that result from the SUMO driver dynamics (reaction time, deceleration phase).

Finally, we wanted to evaluate the amount of time that can be saved through the use of our proposed approach. In this context we have ported the simple crossing scenario (cf. Figure 23) to SUMO, MATSim and to our proposed simulator. To design a *stress test* setting, we have

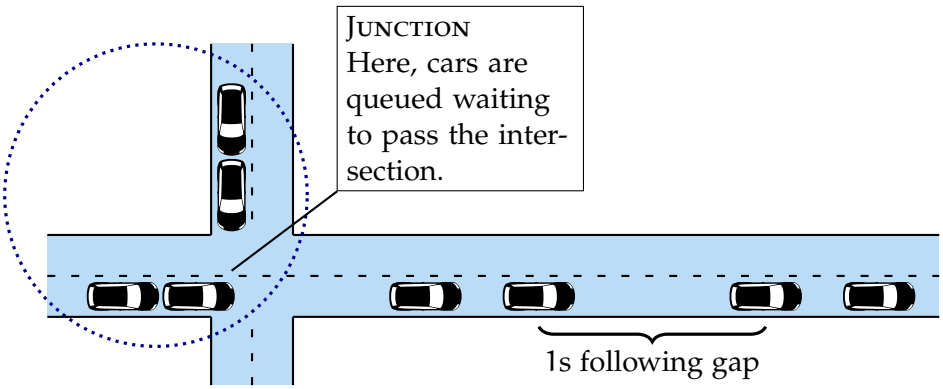


Figure 27: Visualization of the one second gap between the cars in our example scenario.

Table 4: Simulation time for three different simulators.

Simulation engine	Required time for simulation
SUMO	91.98 s
MATSim	2.22 s
Our approach	0.75 s

configured two flows which insert a total of 30000 cars with a gap of 3s between the departure of two consecutive cars on both $road_1$ and $road_2$. Both flows have $road_3$ as their destination.

We have ran the simulation again on our high-end computer with 64 GB of RAM, an Intel Xeon E5-2630 processor and all simulation files stored in a ramdisk to avoid delays caused by slow disk I/O operations. We have used the linux command `time` to measure the runtime. Table 4 shows the runtime in seconds that was required for one full simulation run of this scenario in SUMO, MATSim and our proposed simulator. In this context, the simulations have been performed 10 times and the average runtime value has been taken.

It can be seen that SUMO, compared to the other two, has the longest runtime with 91.98s. MATSim can reduce that time significantly to only 2.22s. Our proposed approach, however, only requires 0.75s. Hence, our approach requires only around 0.8% of the time that is required to perform a simulation in SUMO. The difference between our approach and MATSim (which, in essence, works very similar) can be explained because our approach is tailored to this one specific use case: we avoided many features that MATSim has and we have performed a highly optimized C++ implementation, specifically with the speed optimization in mind.

4.5 CHAPTER SUMMARY

While SUMO is a fully featured microscopic simulation framework it is not suitable for traffic optimization techniques where a large number of simulations has to be performed. This of course applies also to our GA based traffic optimization methodology where each evaluation of a chromosome's fitness requires one distinct traffic simulation. The run times of SUMO simulations are simply too high to get results back in a feasible time. This, again, becomes a problem when trying to optimize traffic online. The reason for SUMOs long execution times lies in the very detailed, agent based vehicle movement model used. This is why it is necessary to move to faster (but also less precise) approaches. In this context it is important, that these approaches are still a good approximation of what we get from a full microscopic simulation.

We have taken a look at MATSim, which relies on Gawron's queue model and introduces a solution to the unfair junction problem described by Gawron. However, it became quickly obvious that the results are not comparable to a microscopic simulator like, e.g., SUMO. One of the major problems is the macroscopic treatment of junctions and therefore the lack of a dedicated junction logic. We have proposed a different approach, which also relies on Gawron's queue model, but allows the implementation of arbitrarily complex junction logics. We have further shown, that this approach is comparable to the results obtained from SUMO and thus a good approximation of a full microscopic simulation.

5

ROUTE CHOICE OPTIMIZATION USING DISTRIBUTED GENETIC ALGORITHMS

5.1 CHAPTER OVERVIEW

This chapter is based on previous work by the author [19, 22] and reflects his contributions to the respective work.

In this chapter we present a novel decentralized route assignment algorithm which is able to assess good route choices by using an island model GA. We aim for a system optimal solution, i. e., finding route assignments for all cars, such that the total travel cost for the system as a whole becomes minimal. To this end, the average travel time of all drivers is taken as the target function for the optimization. The proposed solution encompasses two key components. The first is a decentralized traffic information system: cars exchange information (such as their current position and the desired destination) by locally communicating with other cars in proximity. Cars also propagate information about other cars, so that a picture of the current traffic demand and traffic situation is disseminated. The result is a constantly updated and supplemented snapshot of the current traffic situation in the road network stored at each peer.

The second component is the optimization engine. Its principle of operation is as follows: each car – or, more precisely, the car's local navigation device – constitutes an "island", on which solution candidates (individuals) "live". Each such solution candidate represents a corresponding route assignment for each car. Locally, within each car, the optimization problem based on the local knowledge about the traffic situation is conducted using an own, isolated GA. This will result in individuals which exhibit a good target function value (i. e., short average travel times). In the local optimization, the evaluation of the target function is performed using the light-weight traffic simulation engine introduced in Chapter 4. From time to time, individuals are exchanged between cars – this is the principle behind an Island GA.

The obtained information about favorable route choices is then taken to the real world: each car (re)adjusts its own route choice to match the best locally known individual. Note that this does not mean that all cars will in general follow one identical, optimal global solution. However, we find that choosing the locally optimal solution in combination with exchanging good solutions still results in good route choices. Moreover, the distributed optimization based on continuously exchanged traffic information data is inherently able to quickly react to unexpected changes in the traffic pattern.

Implementing such a joint route choice optimization leads to many interesting problems, which we tackle in this chapter. For instance, not every car is ensured to have the same view on the current traffic situation (e.g. cars have incomplete or contradictory knowledge) which brings many challenges to a distributed optimization.

5.2 DISTRIBUTED GENETIC ALGORITHMS

We have already discussed the general idea behind sequential GAs in Section 3.2: generally, to recall, such algorithms proceed in an iterative manner. The main idea is to generate new populations of individuals from the old ones by applying some variation operators (such as cross-over and random mutation) to find a global optimum. This process requires a large amount of evaluations of the fitness function which rises linearly with every generation that the GA needs to calculate before converging towards a good solution. These evaluations, depending on the size of the underlying road network and the number of vehicles that are being optimized, may become very complex and time consuming. To minimize the computational footprint, researchers began to study different ways to increase the efficiency of GAs. Quickly, several methods of making GAs parallel have emerged [2]. These methods can be put in one of two categories.

The first category is formed by all those methods that work on one single population. In this case the GA itself is not different from the classic sequential solutions. It is more that the implementation is done in a way that the individual genetic operations can be performed significantly faster. This may, for example, include a parallel evaluation or the execution of certain (sub-)portions on the GA on parallelized hardware.

The second category is formed by these approaches that work on multiple populations. These approaches are commonly referred to as parallel GA models (PGAs) [23, 26]. Contrary to the first category, PGAs are not just parallelized versions of the sequential GAs. Instead, PGAs correspond to the development of several, independent populations simultaneously. More precisely, we are dealing with multiple, isolated GA instances each of them performing the genetic operations (selection, cross-over, mutation) on its local population only. Additionally, individuals—preferably those with very good fitness values—are intermittently exchanged between these GA instances. This exchange is called *migration*.

These PGAs try to mimic what we can observe in nature. If we take a look at how evolution works in reality we see that it usually does not operate on a single panmictic population in which a every individual is able to mate with any other individual in the entire population. Instead, it can be observed that evolution, that is selection and reproduction, usually takes place in subgroups or neighborhoods.

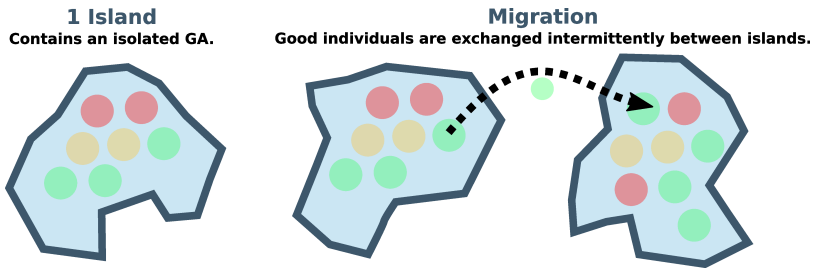


Figure 28: Island GA models.

In fact, using parallel GAs often leads not only to a faster execution time but also to overall better results (for the same amount of computational resources invested) [107]. These improvements are also noticeable when the PGA is executed on a single processor. The theoretical explanation of why distributed GAs do work has been discussed thoroughly in literature [2]. The main reason, in this context, for the increase in performance is the use of the aforementioned structured population, either in the form of a set of islands [98] or a diffusion grid [96].

5.3 PROBLEM FORMULATION AND APPROACH

5.3.1 *Graph Representation and Target Function*

In our proposed approach, the road map again is represented by a graph $G = (V, E)$, consisting of a set of vertices V that represent the intersections, and a set of directed edges $E \subset V \times V$, which describe the roads. Each car c_i is assigned with a source $s_i \in V$, a desired destination $d_i \in V$ and a departure time t_i. The goal of the GA is to find an assignment of a route $R_i = (s_i, x_{i,1}, \ldots, x_{i,n}, d_i), x_{i,j} \in V$ for each of the N cars in $C = \{c_i \mid 1 \leqslant i \leqslant N\}$, such that R_i represents a valid path in G and the global cost function $X = \sum_{i=1}^{N} c(R_i, t_i)$ is minimized, where $c(R, t)$ is the travel time along a route R when starting the trip at time t. The problem is difficult, because the $c(R, t)$ depend on the route choices of other cars.

At any point in time while the system is running on a navigation system, information on other cars will be neither perfect nor necessarily complete. We can therefore, obviously, only aim to approximate a solution to this optimization problem. At any time, the optimization algorithm is running on the subset $\widetilde{C} \subset C$ of cars that are currently driving in the road network. In our evaluation, we investigate the impact of this approximation in relation the original optimization problem with complete and perfect information.

5.3.2 *Information Propagation*

In order to optimize route choices for a given scenario using a GA, the current traffic situation must be known. This includes which cars are present in the network, their current position, their destination and a timestamp of the information. This information is obtained gradually by communicating with the other drivers in the local surroundings. More precisely, each car periodically broadcasts its current knowledge set (which contains information about other cars in the network) appended with its own (current) information at a rate f_B using local wireless communication. This can, e. g., be accomplished using 802.11p, but our system is not limited to this choice. This knowledge set is then received by other cars that are currently within communication range and appended to their local knowledge set. Information about cars which are already known is updated if the received data is more recent.

With an increasing number of cars in a scenario, this opens interesting further research questions related to information aggregation and a more structured broadcast mechanism. This, however, is not in the scope of this chapter.

5.3.3 *Chromosome Modeling*

We now focus on the structure of the chromosome in the individual solutions used in this parallel GA approach. As already described in 3.2, a chromosome constitutes one candidate solution to the problem that the GA aims to solve. In our proposed approach, these candidate solutions correspond to a vector of lists of waypoints: one list with k waypoints for each car about which information is locally known. This relates to assigning each car with a list of intersections in the road network $p_x \in V, 0 < x \leqslant k$ that the particular car visits on its trip from its origin to its destination. The local navigation units are then used to run a simplified, fast traffic simulation in order to assess the quality of each solution candidate by estimating the travel times resulting from the route assignment represented by a chromosome.

In our context, the usage of waypoints has a key advantage over other approaches, in particular the more common method which we presented in Chapter 3 and which works by encoding a specific route by its index within a set of k shortest paths from a car's origin to the car's destination [81, 82]. This advantage of the proposed what we call *VIA point* approach becomes clear when considering multiple GA instances in different cars, which have slightly different information about one particular car. This can be, for instance, in terms of this car's current position. A list of waypoints will typically encode identical or at least very similar routes in all these instances and is thus, in a sense, robust. In contrast, the "third-shortest path" could

mean very different routes for only slightly different starting positions. The waypoint-based route encoding approach therefore enables much easier migration of individuals between GA instances running on different cars.

To avoid confusion due to the different encoding now compared to our evaluation in Chapter 3: we have—on purpose—refrained from evaluating the VIA point approach in the case of the global GA. The reason is straightforward: the global-knowledge GA tries to find a good set of route choices for all cars in the traffic scenario *a-priori*, that is, before the first car enters the road network. If cars were assigned with a VIA point (instead of one of k alternative routes), then they would approach it directly from their origin. This is clearly not the case for our distributed GA approach, where cars perform the optimization while they are driving. As the GA needs some time before it comes to a solution that is good enough, this would result in VIA points set in the middle of the journey and so approached from a different (non deterministic) position. The different nature of these two GA approaches would therefore not allow a valid comparison of the VIA point approach.

The waypoint lists themselves are encoded in a bit field, which constitutes the individual's chromosome. Every entry in a chromosome is called a gene. Every car corresponds to k genes on the chromosome that hold the value of the k waypoints which the car must visit in one particular solution candidate. A car can, in effect, be assigned with $l < k$ waypoints by setting the remaining $k - l$ waypoints to the car's destination.

At this point it should be emphasized that, due to the differences between the knowledge sets, the locally solved optimization problems are generally not identical, but only *approximately* identical. This is a distinguishing property of our approach, which is typically not the case for Island GAs, and which induces a number of interesting challenges. In a sense, our optimization approach builds upon the conjecture that good solutions in the local GA instance of one car will, despite the differences between knowledge sets, also constitute interesting solution candidates for other cars' instances. That is, we assume the optimization problem to be sufficiently smooth that a) migration of individuals between islands (with appropriate adjustments as described below) makes sense, and b) sensible global behavior of the system is achieved if each car acts according to its currently best local solution.

As an immediate consequence of the differences in the knowledge sets, the set of cars considered in the optimization and also the position of the information about one specific car on the chromosome is not necessarily the same across different islands. Therefore, whenever individuals are exchanged between islands, the required index

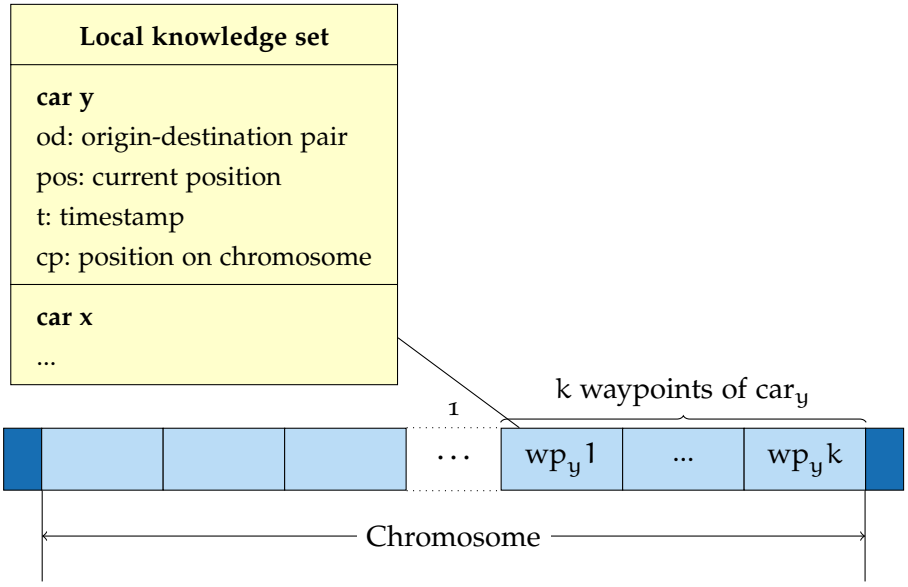


Figure 29: Illustration of chromosome and knowledge set.

information is added, which then allows for the combination of individuals from different islands.

With respect to the selection of waypoints, note that a given area in a road network does not necessarily need to constitute a connected graph, i. e., a graph in which every vertex can be reached from all other vertices. While it is trivial to model artificial road networks using connected graphs, this does not apply for scenarios which are provided by external suppliers taken from empirically collected data (as used in our evaluation). To this end, it has to be ensured that each waypoint $p_k \in V$ of car c_k is valid for its given OD-pair (s_k, d_k). By "valid" we mean that p_k can be reached from s_k and d_k can be reached from p_k. Furthermore, it might be wise to limit the set of valid waypoints even further to avoid a too large search space. In large road networks, for example, waypoints that are located far off a vehicle's origin and destination are very unlikely to result in reasonable solutions. In this case, it is conceivable to require valid waypoints to be located within a certain area around both the source and the destination point.

The chromosome itself is stored independently from the knowledge set (cf. Section 5.3.2). This simplifies the implementation, because the information stored in the knowledge set is not part of the genetic optimization (that is, it is not modified by the genetic operators). The information from the knowledge set is, however, required in order to assess the quality of a solution candidate. Moreover, the chromosomes of the currently running GA instance have to be updated whenever new information about the current traffic situation is learned from an incoming beacon. If the presence of a particular vehicle is recognized for the first time, all chromosomes have to be

extended to contain an entry for that car. Due to the absence of information about good waypoints for newly added cars in the context of what is encoded on the local chromosomes, these cars start with no waypoints assigned (that is, they take the shortest path).

When cars have passed one of their waypoints, the chromosomes are adjusted to remove the respective waypoint. This is necessary to avoid oscillations where cars that already have passed one particular waypoint are told to revisit it. That may happen as the best chromosome—the one that is used for routing—most likely still is the best solution *after* the passage of the waypoint, i.e., the way back to the waypoint and then proceeding to the destination may still be better than the alternative route assignment variants that are encoded on the other chromosomes in the population. This behavior, however, is not tolerable from the driver's point of view. To this end, we cycle through all local chromosomes and remove that waypoint from the related car's k waypoint list. In order to decide whether one particular car has passed a waypoint or not, we use the information about the current position from the last beacon as well as the car's current waypoint from the local knowledge set. To this end, we assume the car has passed its via point when it is located on the shortest path between its via point and its desired destination.

Right now, this fixes the chromosomes which include the cars' current via point and leaves the other chromosomes untouched. This again can cause the PGA to perform slowly: the reason is that legitimate via points on other chromosomes may become fairly bad while the car proceeds on his way to the destination point. Waiting for the GA to level out these bad via points may take very long and cause avoidable slow downs. Additionally to the aforementioned adaptation of those chromosomes that hold the current via of one car, we suggest to adapt *all* via points of *all* chromosomes—not just the own ones. To this end, we suggest to apply the mutation operator to each via point that has a greater distance from the cars' current position than has the destination point—let us call such via points plausible via points. Here, we have to differ between the plausibility and the validity of via points. Invalid via points are points that cannot be reached from the current position or do not allow to reach the destination once passed. Implausible valid points, on the contrary, can be valid but do not reflect a reasonable solution from the drivers' perspective. Practically, the elimination of implausible via points can be achieved by cycling through all via points on all chromosomes and checking their distances based on the cars' last known positions in the local knowledge set. All via points that are further away than a threshold value are marked implausible and mutated. This threshold value can be designed elastic, e.g., via points must not increase the total travel distance between the current position and the destination by a factor f. The mutation operator itself is advised to only give back

plausible via points. This way we can ensure that, at no point in time, there is an implausible via point on any chromosome.

Finally, when cars have reached their destination, it is no longer required to account for them in the optimization. These cars' entries are therefore removed entirely from all local chromosomes.

5.3.4 *Genetic Operators*

5.3.4.1 *Selection and cross-over*

The selection of individuals is performed using the rank based method as it was described in Section 3.3.5. The pairing of two selected individuals is achieved by the *cross-over* operator. This operator, in essence, takes the "genetic material" of two individuals selected by the roulette wheel and combines them to a new individual. Here we use the so-called *uniform cross-over* operator. That is, assume a chromosome holding information about N cars, each being assigned with k waypoints. Then, given two individuals I_i and I_j chosen for pairing, a new individual $I = \{i_1, \ldots, i_{N \cdot k}\}$ is formed as follows. For each gene i_x at position $1 \leqslant x \leqslant N \cdot k$ the value of either the first parent's gene $I_i[x]$ or the value of the second parent's gene $I_j[x]$ is assigned with equal probability.

5.3.4.2 *Mutation*

The mutation operator is essential in our proposed approach. One of the problems of GAs is that the evolutionary process may converge too fast to a solution which causes it to be stuck in a local optimum. This problem can be overcome by continuously ensuring sufficient diversity in the candidate solutions with the mutation operator.

We use a static mutation rate p for each waypoint on the chromosome; such a block corresponds to one waypoint of one particular car. In this context, it has to be ensured that after a mutation of one waypoint, the resulting new waypoint remains valid for the respective car (cf. Section 5.3.3). When a mutation occurs, a waypoint is therefore randomly chosen from the set of all valid waypoints.

5.3.4.3 *Migration*

The final key operation in an Island GA is the migration of individuals. Migration essentially means that chromosomes from different GA instances (i. e., islands, cars) are exchanged. Because, as explained above, the local knowledge sets and therefore the locally considered optimization problems will differ, the migration process requires special treatment.

The chromosome can only be interpreted in combination with the knowledge to which cars each individual gene refers, and what the current positions and the destinations of these cars are. Therefore,

chromosomes which migrate from one island to another are always broadcast together with the matching local knowledge set. On the receiver side, the received knowledge set is processed first. As described above, this includes that local chromosomes of the receiver are adjusted to the new knowledge set as required. Subsequently, it may only happen that the migrated chromosome is missing some cars from the local knowledge set at the receiver side. In this case, genes for the missing cars are inserted into the immigrant chromosomes, with the waypoints all set to the respective car's destination vertex. The latter is equivalent to having the new cars all take the shortest path without any detours.

The migration strategy itself is an important parameter on the performance of distributed GAs and consists of multiple parameters [24]. These parameters include the migration frequency f_{MIG} (how many generations pass before a migration takes place), the migrants' selection strategy (which individuals migrate), the replacement policy (which individuals get replaced by immigrants) and the migration topology (which islands exchange migrants). The topology in our case is determined by the communication system: individuals migrate between cars which hear each other's respective transmissions. This results in a *Dynamic Nearest Neighbor Topology*. For the migrant selection strategy we follow the elitism approach: the best individual of a population is picked for migration and replaces the worst individual of the other population.

A migrant is transmitted with every broadcast of an information beacon. However, not all incoming migrants are necessarily integrated into the local population, because this would result in too high dynamics. On the receiver side, an immigrant is added to the local instance of the GA whenever a certain minimum number of generations f_{MIG} has passed since the last immigration. Otherwise, received migrant chromosomes are discarded. Note that this is independent from the processing of incoming knowledge set updates, which are always processed.

The authors of [71] investigated the impact of different epoch lengths (number of generations between migrations) on the general quality of distributed GAs. Their general conclusion is that too short epoch lengths (i. e., too many migrations) are counterproductive. They recommend infusing new genetic material into a sub-population only if the sub-population starts stagnating. There is, of course, no fixed number of generations before stagnation occurs, this highly depends on the specific situation. We therefore employ an adaptive approach for setting the migration frequency using the techniques described in [70].

5.3.5 *Route Choices*

Whenever approaching an intersection, the car must take a look at the current state of the GA. In this context it picks the best individual of the current generation and extracts its own set of waypoints from this individual. Then, the car calculates a route from its current position to its destination visiting all waypoints in the order they appear on the chromosome by using a shortest path algorithm between each consecutive pair. When the car again approaches an intersection and the best individual is still the same, it does not adapt its route. However, if the best individual has changed, the route is recalculated.

One issue that may arise from such a design are bad routes while the GA is in an early state or not much information was collected about the situation in the road network: the individuals may then constitute mostly random solutions. We avoid this problem by the way we choose the initial starting population for a GA instance. The fact that we assume that all cars take the shortest path in the initial route assignment in combination with elitism prevents our algorithm to detour cars on random routes when either the local knowledge is too scarce or the GA is in a too early state. To put it differently, in such constellations the shortest path assignment will automatically be the best individual in the current local generation, and will thus be picked for the local routing decision.

5.4 EVALUATION RESULTS

Even though simulation models typically only approximate the behavior of a real system, a realistic traffic demand plays an important role in the evaluation of systems related to vehicular road traffic. Hence, for the evaluation in this chapter, we use a real world scenario from the city of Bologna [12] which is depicted in Figure 30 and represents an observed peak hour traffic demand (8:00 am – 9:00 am). This scenario was originally developed by iTetris [86] and later made publicly available [12]. During the simulated time period, a total of 11000 cars is inserted at a steady rate into the road network.

Usually a traffic simulation starts with an idle network. Therefore it is required to let pass a sufficiently long warm-up period to allow traffic conditions to stabilize before starting the evaluation. In this context, the warm-up period should be long enough such that traffic is getting onto all, or the majority, of the links in the network before the main simulation starts. Figure 31 shows the number of running vehicles in the road network over time. It can be seen that the warm-up phase can be considered finished at 8:15 am when the traffic load in the network has stabilized between 700 and 800 cars. Also the last 15 min are not accounted for in this evaluation as the traffic load starts decreasing.

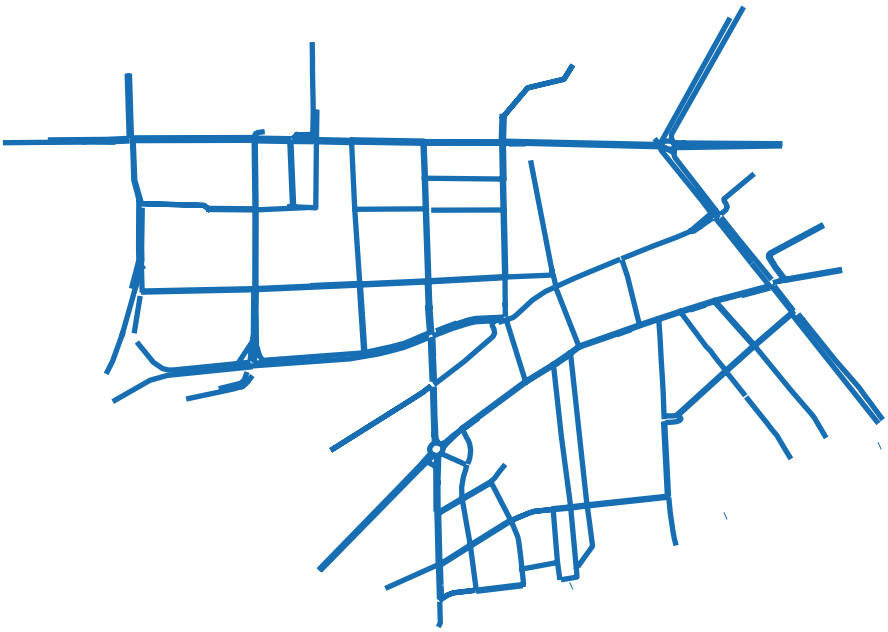


Figure 30: The road network from Bologna.

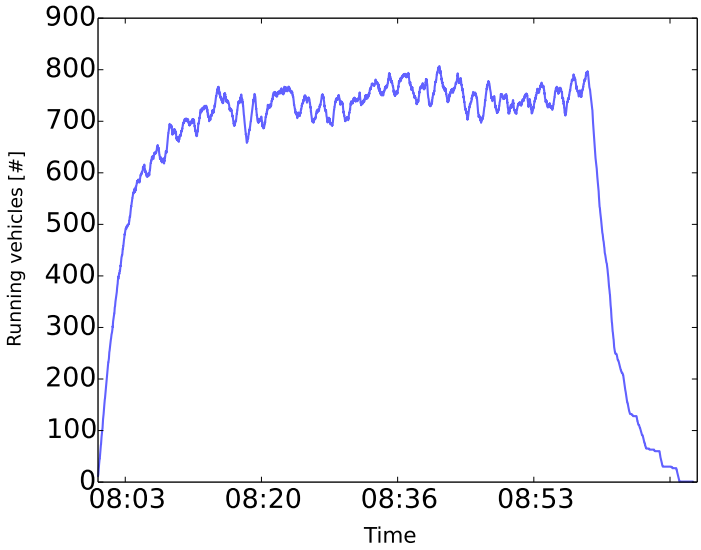


Figure 31: Running vehicles over time.

The evaluation was performed in Veins 3.0 [102], which is an open source framework for running and evaluating vehicular network simulations. Veins links two well-established simulators: OMNeT++ [101], an event-based network simulator, and SUMO [10]. The communication between vehicles was implemented on Veins' 802.11p layer. We used the path loss radio propagation model [84]. Furthermore, we set the TX power to 10 mW, the sensitivity to −89 dBm, the center frequency to 5.890 GHz and the thermal noise to −110 dBm. This resulted in an approximate transmission radius of 300 m. A lower transmission radius of 175 m or 50 m had no significant impact on the outcome of our experiments.

To avoid confusion, please note that our evaluation scenario here includes *two different traffic simulation engines*: first, the Veins-based simulation in which the cars move according to SUMO's microscopic model, and in which OMNeT++ simulates the communication channel for exchanging vehicle information and GA individuals. This Veins simulation corresponds to what would be the "real world" is meant to approximate the real movement of the cars. Second, each of the cars in this "real world" locally runs a GA instance. And this local GA instance uses a *different* kind of simulation to assess the fitness of the local individuals. These simulations uses the simplified, queue-based model described above in Chapter 4 to estimate the approximate travel times for different solution candidates. This other simulation would in the very same way be used for the local optimization process in each navigation system in a real-world deployment of our system. The simulation models differ: the queue-based movement model is much less sophisticated than SUMO's, just like in a real-world application the queue-based model is not identical to the car's movement in the real world. The impact of these deviations, caused by the simplifying assumptions made in the queue-based local simulation model, are therefore inherently accounted for in our evaluation results.

Microscopic traffic simulation models use random variables and sample from random distributions to represent the dynamic decisions made by the simulated agents. This in the first place includes the drivers' non predictable behavior such as dawdling or their reaction time. In order to obtain statistically sound results, we repeat all experiments five times; the number of repetitions was obtained using the method described in [106]. Where applicable, we include error bars that represent the 95 % confidence interval in our plots.

Each GA instance (that is running on the navigation devices) was configured with a fixed sub-population size of 20 individuals. The choice of 20 individuals per sub-population ensures both enough diversity on each island [71] and a reasonable computation effort per generation. Note, that in our distributed GA instance *every* car is equipped with 20 individuals, so the entire distributed GA instance

is effectively working with a multiple of that. Additionally, each instance is configured to use *elitism*. In this context, 3 of the fittest individuals are moved to the next generation unchanged. According to [75] a reasonable mutation rate for a variety of problems is $\frac{1}{l_C}$, where l_C is the chromosome length. While a too high mutation rate can turn the search into a primitive random search, it was necessary to deviate from this suggestion. The initial population in GAs is usually picked randomly and thus can be assumed to have enough diversity. In our case, however, it is not recommended to send cars on random paths in the early stage of the GA. Assigning each (new) car on the chromosomes with an empty list of waypoints makes the population homogeneous and the crossover operator can only recombine alleles that exist in the population.

In order to quickly leverage the diversity and accelerate the search process a higher mutation rate is required. We have performed a small experiment in the setting described above where we tried out several different mutation rates to see which one works best for this particular use case. In this context, we have recorded the diversity of the population at one particular car for 50 generations and for mutation rates $m \in \{1\%, 2\%, 5\%, 10\%, 15\%, 20\%, 25\%\}$. The size of the local knowledge set at that car had size 1000 and we used only one VIA point per car and per chromosome. That is, the chromosome held 1000 mutable alleles. Migration has been specifically turned off to avoid any external influences. The population size was configured to be 20. We have performed an analysis of the average Hamming distance by interpreting the chromosomes as bit strings of length 8000: the Bologna scenario has 159 junctions that can be used as VIA points; each allele therefore can be represented by at most 8 bits.

Figure 32 now shows what value the average Hamming distance converges to when applying the various mutation rates described above, all starting with a entirely homogeneous population.

In order to find out which of these is the "best diversity" for our scenario, we have used the contribution metric as introduced by [58]. The contribution is defined as the ration of the number of successful crossovers, i. e., those that produce at least one child who is better than both parents in terms of the fitness value, and the number of all crossovers. In this context we want to target a diversity which causes the *contribution* to be particularly good. Figure 33 shows the contribution for 100 randomly shuffled populations (again, with 20 individuals each) and with varying diversity between 0 and 4000. We have chosen 4000 as the maximum as it constitutes the worst case, i. e., a fully random population. It can be seen that the contribution is high as well as very stable for a diversity described by an average Hamming distance between $800 - 1500$. As a mutation rate of 10% seems to converge to exactly this range, we consider it a good choice (in this scenario) and use it in the remainder on this chapter.

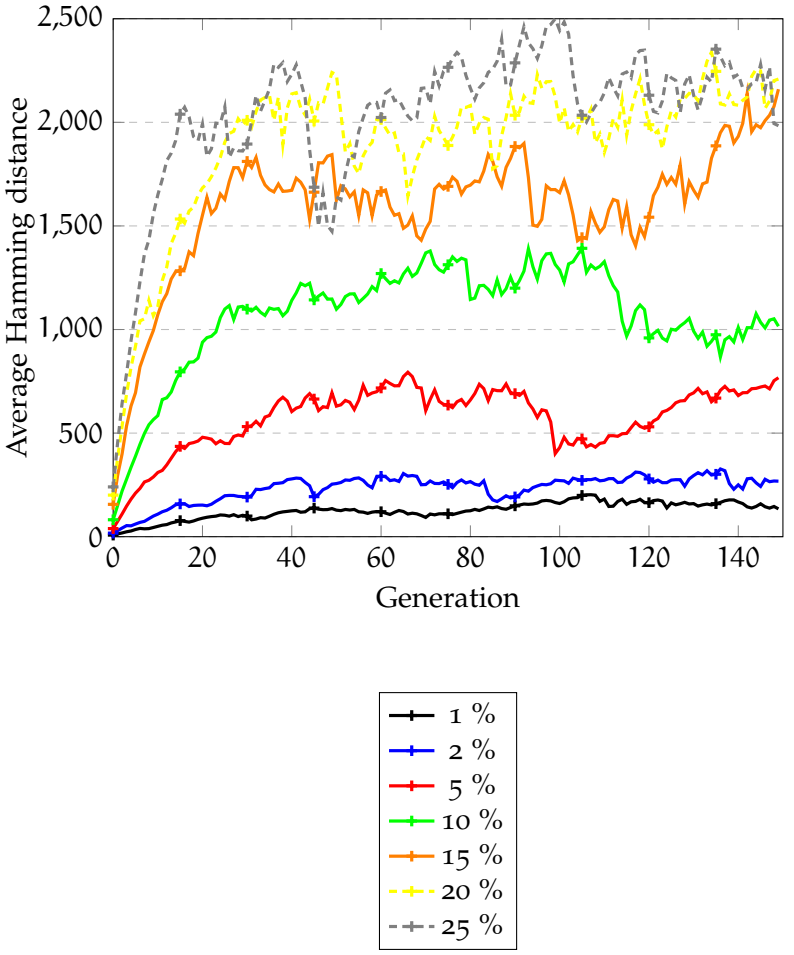


Figure 32: Average Hamming distance for different mutation rates.

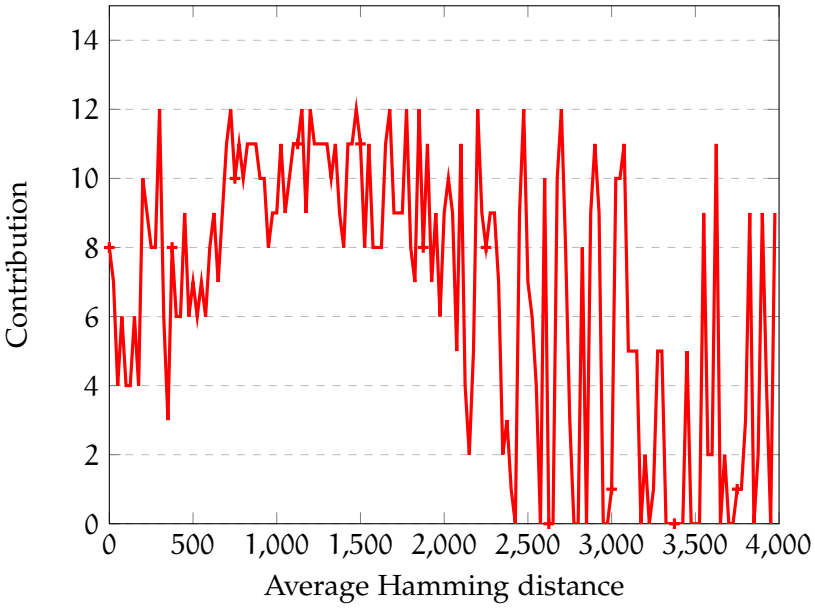


Figure 33: Contribution in relation to the diversity of the population.

Table 5: Average travel time for different approaches and relative improvements compared to the unoptimized case.

Methodology	Average travel time	Gain
(unoptimized case)	303.45 s	0.00%
FCD on central server (0s lag)	283.21 s	6.67%
FCD on central server (60s lag)	285.29 s	5.99%
FCD on central server (120s lag)	285.30 s	5.98%
FCD on central server (180s lag)	286.40 s	5.62%
FCD on central server (360s lag)	291.01 s	4.10%
FCD distributed approach	290.66 s	4.22%
Our distributed GA approach	242.28 s	20.16%
Global knowledge GA	231.55 s	23.70%

Coming back to the original experiment: in the unoptimized scenario an average travel time of 303 s can be observed. We evaluate our proposed route choice optimization algorithm in this scenario and compare the effectiveness of the proposed algorithm to a number of existing route assignment solutions. The first approach we compare to is the use of floating car data (FCD) stored on a central route computation server. In this case, speed and position information for each vehicle is transmitted to a traffic center. This central database is queried by all vehicles periodically in order to adjust routes based on the current view of the traffic situation. In this context, we assume that all vehicles adjust their routing decision the moment an updated information is available on the server. Collecting floating car data (e. g., using detector loops or mobile phone traces) is what most of the present systems do (cf. Section 2.6). We also evaluate the case where the data on the central server is $t \in \{0, 60, 120, 180, 360\}$ seconds old. The second approach we compare to is the decentralized use of floating car data. Here, the speed and route information of each vehicle is transmitted to other vehicles that are in the local surroundings. Cars learn about the traffic situation gradually and calculate their best route based on their local view of the traffic situation. To have an upper limit for optimization, we also performed a comparison to a centralized, global-knowledge GA as described in Section 3. As discussed before, perfect knowledge about all past, present and future trips and a central optimization of all routes based thereupon is of course unrealistic in the real world.

However, it is doable in a simulation were all cars along with their departure times and OD-pairs are known in advance. The comparison against a global-knowledge GA hence allows for an insight how

much our proposed approach lags behind this idealized solution. Figure 5 shows the results of these evaluations. While approaches targeting a user equilibrium improve the travel times by 4–7 % relative to the unoptimized case, our proposed system optimal distributed GA yields an improvement of 20 %.

The idea of the system optimum is that all entities cooperate in the favor of the system as a whole. That is why our proposed algorithm not only relies on a view of the current traffic situation that is as accurate as possible but also on a preferably high number of vehicles that participate. However, it cannot be foreseen how many vehicles will actually adopt this technology—specially in the early bootstrapping phase when the product hits the market.

In order to get a feeling for how our route optimization algorithm performs under the premise of different penetration rates we have conducted an experiment where we have measured the average travel time of all vehicles for penetration rates between 0% and 100% with a step size of 10%. The average travel time, in this context, is calculated over all vehicles in the network. The reason is because we not only want to reflect the improvement (or disimprovement) of the cars that "participate" but also reflect potential effects of those cars' decisions on the other offline cars in the network. It does not help if the improvement of those cars that participate comes at huge disadvantages for the rest of the drivers.

The evaluation has been carried out analogously to Section 5.4 with the difference that only a certain percentage of cars was equipped with our optimization application. The others had no ability to communicate nor did they perform any kind of optimization. Hence, they were not part of any chromosome or local knowledge base. This means, that the simulations which are needed to evaluate the fitness value of each chromosome did not include any cars that were not equipped. Routing decisions were solely made based on the reality that was reconstructed from incomplete knowledge. However, and this is important to make clear, albeit the unequipped cars did not appear on any chromosome or simulative fitness evaluation, they surely were driving through the simulated "reality" being properly accounted for in the calculation of the average travel times.

The results that are summarized in Table 7 show that a decreasing penetration rate indeed has an impact on the quality of the resulting traffic situation: a penetration rate of 90% alone causes the gain (to recall, in terms of a lower total travel time for the whole system) to drop from 20.16% to 17.21%. The gain can go as low as 4.85% for a penetration rate of 10%. Still, even with a penetration rate of only 10% we can compete with, and with a penetration rate of only 20% we can outrun all FCD approaches which have a gain of 4.10%-6.67% in the case of full penetration. One important thing that we also see is that

Table 7: Impact of different penetration rates on the quality of the result.

Penetration rate	Average travel time	Gain
0%	303.45 s	0.00%
10%	288.73 s	4.85%
20%	278.80 s	8.12%
30%	272.80 s	10.10%
40%	262.20 s	13.59%
50%	258.15 s	14.92%
60%	256.02 s	15.63%
70%	255.26 s	15.88%
80%	254.11 s	16.25%
90%	251.21 s	17.21%
100%	242.28 s	20.16%

our approach does not cause the traffic conditions to worsen at any level of penetration.

Figure 34 shows the cumulative distribution function of the relative changes in travel time for all individual vehicles in the case of full penetration. The relative travel time change is the ratio between the car's actual travel time and its travel time in the unoptimized case. Due to the similarity of the results for different lag times in the central FCD approaches, we exemplarily include only the central FCD approach with a lag of 120 s in our plots for better readability. It can be seen, that approximately 70 % of all cars in the scenario are able to improve their travel time using our proposed approach. For most of the remaining cars, the travel time does not change significantly, except for a relatively small group which has to take the disadvantage of a longer travel time in favor of the system as a whole. This is expected behavior in a system optimal route assignment. The FCD approaches in contrast result in a broader diffusion of gains and losses: a larger number of cars improves the travel time but also a larger number ends up taking a longer detour.

Figure 35 shows the cumulative distribution function of the route lengths in the scenario for the evaluated algorithms. In can be seen that, with the exception of the idealized solution from the global-knowledge GA, our proposed approach results in the least impact to the route lengths.

Figure 36 depicts the route length in relation to the observed relative change in travel time for that particular route for our proposed approach. While gains and losses in travel time are distributed evenly for shorter routes (< 1000 m), this is not the case for longer routes. It

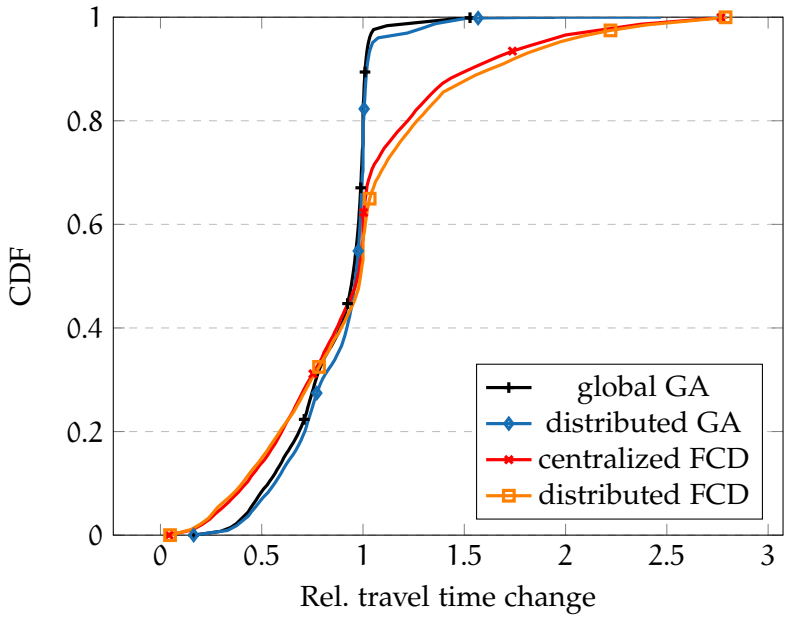


Figure 34: CDF of relative travel time changes.

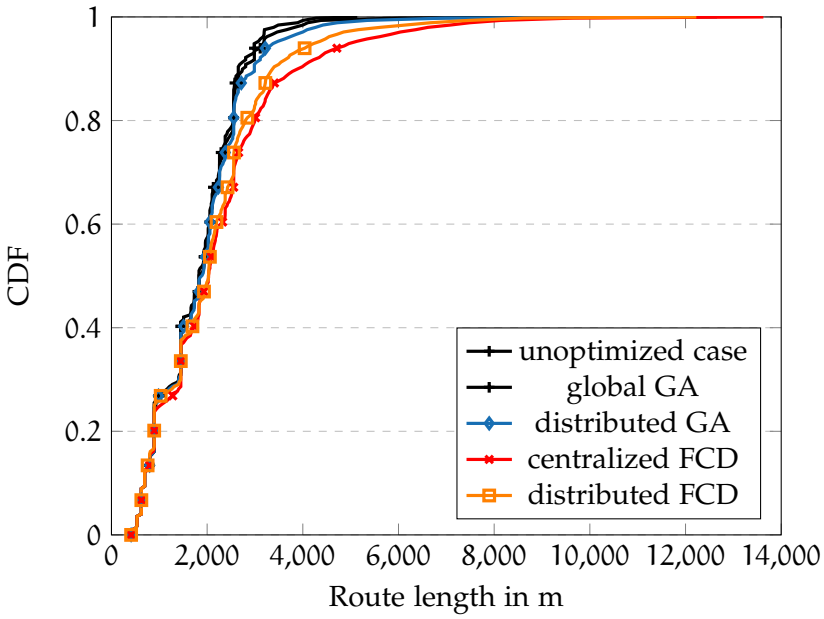


Figure 35: Distribution of route lengths.

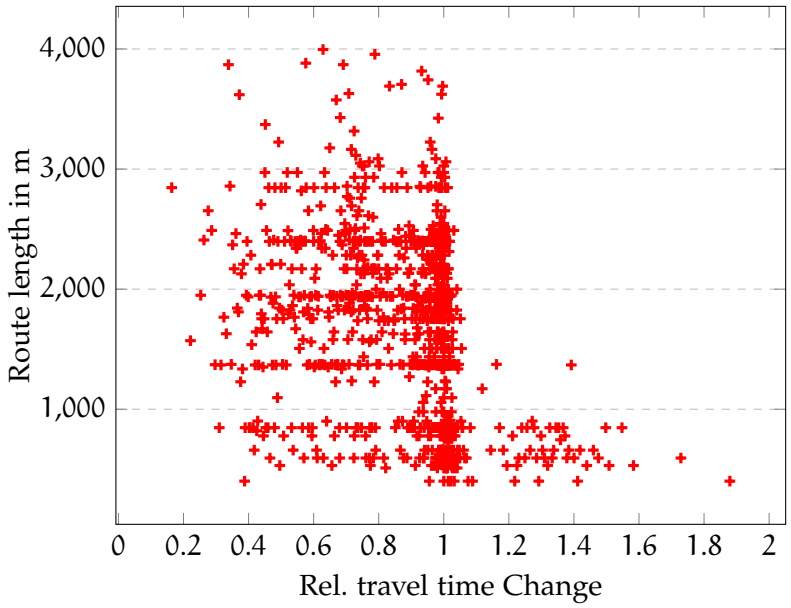


Figure 36: Correlation of route length and gain.

may thus be assumed that the chance to benefit from our proposed algorithm increases with the distance traveled by a specific car.

5.5 INFLUENCE ON EMISSIONS AND FUEL CONSUMPTION

Aside from showing that our proposed approach in fact does improve the road network efficiency, it can be worth to take a look at the environmental impacts they come with. In this section, we focus on the evaluation and quantification of these impacts in terms of CO_2 (carbon dioxide) emissions as well as the fuel consumption in the same same scenario as above. In this context, we compare two sets of *route choices* in the given scenario. To recall: a set of route choices in essence describes which route *each* of the 11,000 vehicles in the Bologna scenario from Section 5.4 drives from his origin to his destination. On the one hand we have the *unoptimized* route choices: every driver chooses the shortest path (in terms of the free-flow travel time) from his origin to his destination. We compare this to *optimized* route choices obtained by using the optimization approach as it is described in Section 5.4.

The emissions themselves are recorded using SUMO's on-board tools: SUMO comes equipped with the ability to record emissions and fuel consumption according to the HBEFA v2.1 model [50]. This model makes a few abstractions from the fuel consumption observed in the real world, e. g., it only uses the vehicles' class (up to Euro 5), speed and acceleration but entirely ignores the slope of the road. Even though the model supports 56 different vehicle classes, most realistic traffic scenarios have either no information concerning the vehicle

classes, or they cluster cars in just a few of them. In the used Bologna scenario, e. g., the authors only differ between heavy duty vehicles (HDV) and passenger cars. This has to be kept in mind, when later drawing conclusions from the simulated model to the real world.

The emissions are recorded in $\frac{\text{mg}}{\text{s}}$ and the fuel consumption in $\frac{\text{ml}}{\text{s}}$ in each timestep for each car. Additionally, the cars' total emission and fuel consumption is recorded after they have finished their journey. The two metrics are closely related, but not fully equivalent, because different classes of vehicles use different types of fuel, which in turn correspond to different amounts of CO_2 per litre of fuel [50].

It can be observed that the route choice optimization increases the total emissions of CO_2 from 2605.43 kg to 3000.96 kg (+15.18 %) and the total fuel consumption from 1038.74 l to 1196.36 l (+15.17 %), while the total driven distance increases from 13 075.66 km to 15 977.07 km (+22.18 %). However, due to the optimization, a different set of vehicles will reach their destination during the simulated time frame. More precisely, 8556 vehicles arrive at their destination before the route choice optimization is applied. Using the optimized route choices this number grows to 9369. Therefore, the absolute fuel consumption or CO_2 emission do not constitute suitable metrics: they do not refer to the resources spent for achieving the same goal or delivering the same amount of service.

We therefore compare the additional expenses of the optimization in terms of higher CO_2 emissions and a higher fuel consumption per kilometer of the *unoptimized* route. That is, in a sense, we use the unoptimized (i. e., shortest path) route length as a measure for the "amount" of transportation service delivered to the driver of the respective car. Let r_c be the route of car c as it was planned in the unoptimized scenario and l_{r_c} the length of this route. Let $CO_2(r_c)$ and $\text{FUEL}(r_c)$ denote the CO_2 emission and the fuel consumption along that route, respectively. Then, given the CO_2 emission $CO_2(\widehat{r_c})$ and the fuel consumption $\text{FUEL}(\widehat{r_c})$ in the optimized scenario, we normalize these values to the length of the unoptimized route l_{r_c}. Furthermore, we are interested the in the ratios $\frac{\text{FUEL}(\widehat{r_c})}{\text{FUEL}(r_c)}$ and $\frac{CO_2(\widehat{r_c})}{CO_2(r_c)}$ for all cars that finish their journey in both the unoptimized and the optimized case. These ratios show by which factor the CO_2 emission and the fuel consumption of a vehicle driving from the same origin to the same destination have changed.

Figure 37 shows the cumulative distribution function (CDF) of the CO_2 emissions per kilometer before and after the optimization of the route choices, as discussed before in both cases in relation to the length of the unoptimized, shortest-path route. It can be seen that the values do not change much: only a small subset of vehicles experience noticeably higher emissions and a higher fuel consumption. Also, only a small number of vehicles can lower these values to a non-

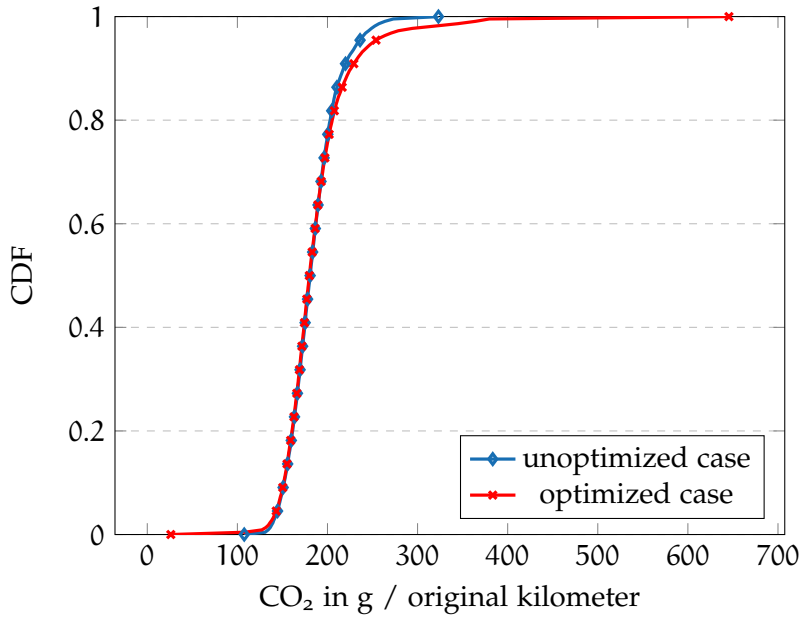


Figure 37: Comparison of the CO_2 emissions.

negligible extent. The same pattern can be observed when looking at the comparison of the absolute fuel consumption in Figure 38.

While the CDFs of the absolute values show that the distribution of emissions and fuel consumption do not change much, it does not show what that means from the drivers' point of view. Figure 39 shows the CDF of the relative changes in the absolute CO_2 emissions and fuel consumption per original kilometer after the optimization has been applied. Here, it can be seen that, around 40 % of all vehicles can reduce their CO_2 emission and fuel consumption noticeably while the other 60 % experience increased values. This indicates that disadvantages (in terms of a non-optimal emission / fuel consumption) are reallocated among the cars: we still have cars that cause much pollution, its just that these are different cars after the optimization of the route choices. In a next step, we take a look at the same values, but now accumulated over all routes. It can be seen that the total emissions of CO_2 have increased from 199.25 g to 207.33 g and the fuel consumption has increased from 79.44 ml to 82.65 ml per kilometer. This corresponds to a 4.05 % increase of CO_2 emission and an increase of 4.04 % in the fuel consumption for the entire system. The observed increase in emissions and fuel consumption after applying our proposed route choice optimization methodology is relatively small given the fact that the costs for the system as a whole (in terms of the total travel time) can be reduced by over 20 % [22].

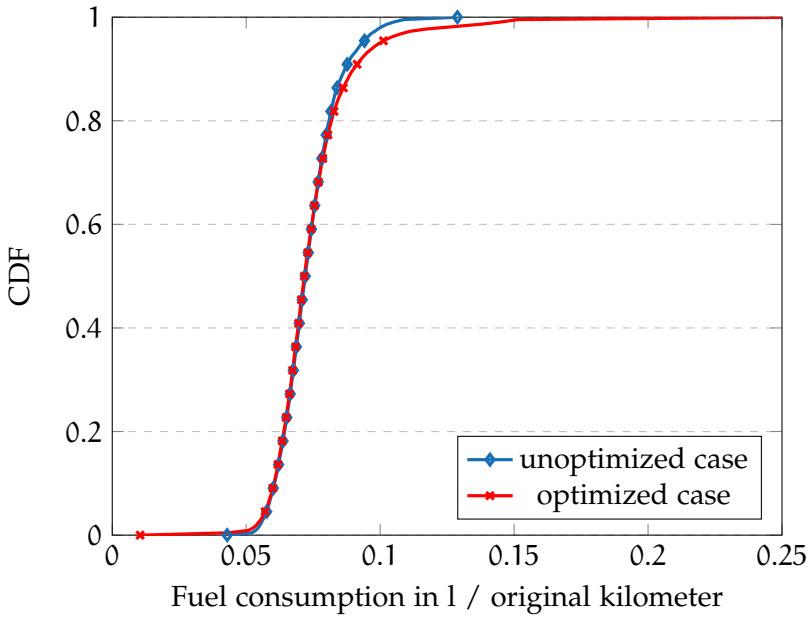


Figure 38: Comparison of the fuel consumption.

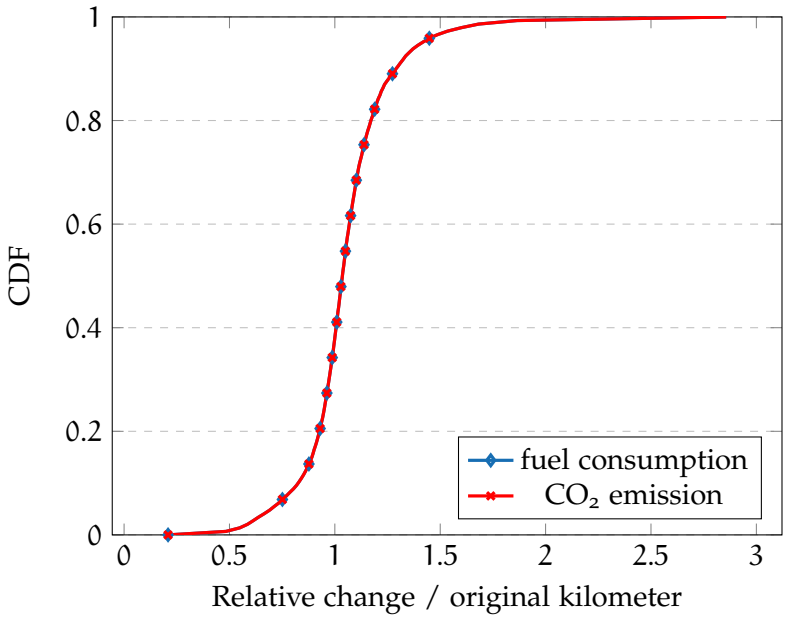


Figure 39: Relative change after optimization.

5.6 APPLICABILITY TO VARIABLE INFLOW

5.6.1 *Definition of Variable Inflow*

As described in Section 5.4 we have evaluated our distributed GA approach in a scenario, in which the cars are added at a constant rate. However, a constant inflow rate that remains stable over a longer time frame cannot be assumed in realistic environments. In this section we aim for evaluating the applicability of our approach in the case of a variable inflow.

In a first step, it is required to define the term *variable inflow*. A variable inflow, in this context, means that the rate at which cars are added into the scenario varies over time—instead of being constant. An Inflow function, in general, is a function which for every time slot t (of a certain, predefined length) gives back the number of cars that entered the scenario per time-frame. This time-frame usually is chosen to be one hour.

Taking a look at Figure 40, we see a static inflow function $f(t) = Q_0$ and an alternative, variable inflow function $g(t)$ which inserts the same amount of cars during the entire observation time span T. Without loss of generality, we limit the analysis that we conduct in the following sections to cosine-shaped variable inflow functions of this form:

$$g(t) = Q_0 - \Delta Q \cdot \cos\left(t \cdot \frac{2 \cdot \pi \cdot \psi}{T}\right) \tag{14}$$

This type of function in essence oscillates around a base value Q_0 with a cosine function with an amplitude of ΔQ and a frequency of $f_{freq} = \frac{\psi}{T}$. In this context T denotes the observation time frame which in essence is the duration of the entire simulation.

Returning to our variable inflow function: we only aim for changing the pattern in which cars are inserted into the scenario and not the total number of cars inserted. That is, we require that the total number of cars inserted by a new inflow function $g(t)$ in the entire observation time frame must be equal to the original number of inserted cars. More formally that means that the following equation must hold:

$$Q_0 \cdot T = \int_0^T g(t)\,dt \tag{15}$$

As we are interested in the behavior of our optimization approach under continuously changing traffic inflow functions. Hence, we will vary parameter ψ which, to be more precise, allows us for evaluating the efficiency of our distributed GA approach for variable inflows of different frequencies. In essence, the parameter ψ describes how many full cosine periods the cosine inflow function will make during the whole simulation with duration T.

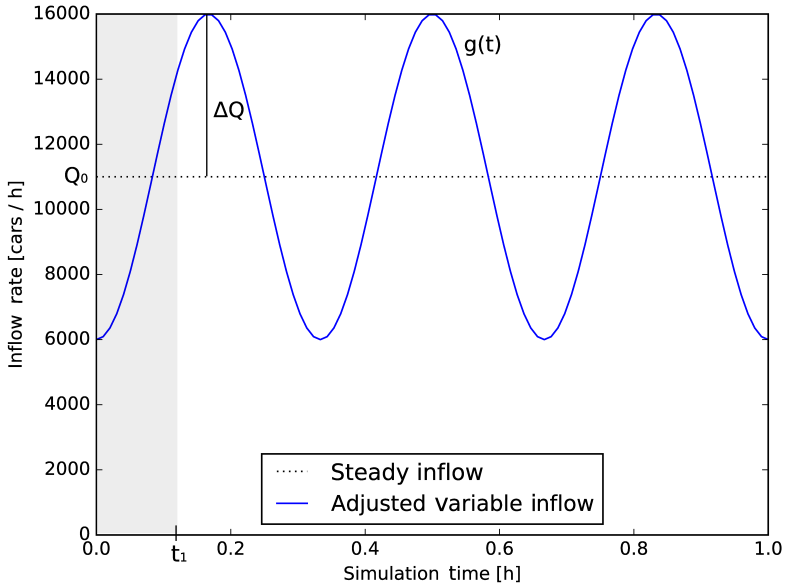


Figure 40: Constant vs. variable inflow.

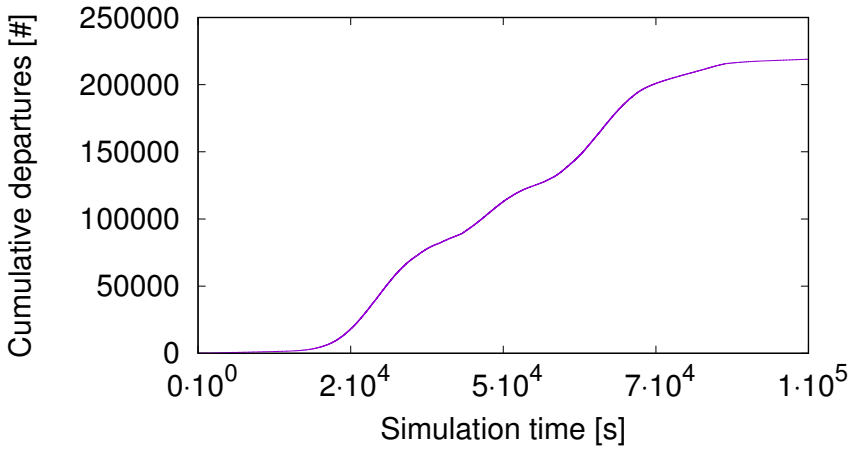


Figure 41: CDF of the departures in the LuST scenario.

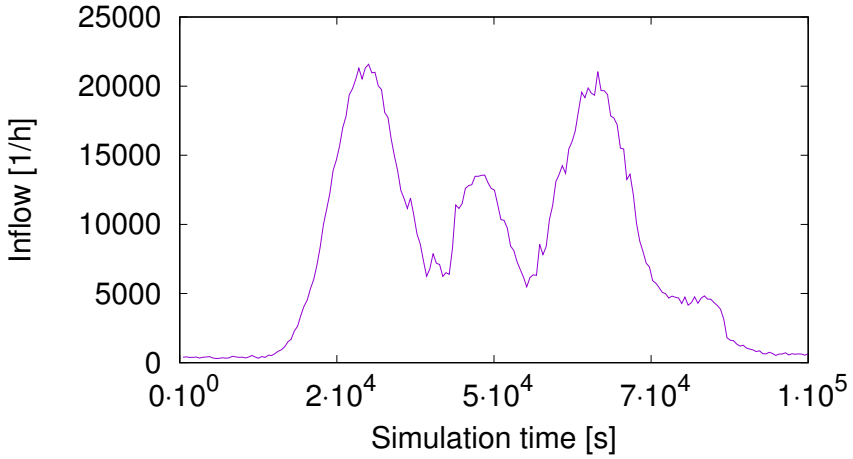


Figure 42: Traffic inflow function in the LuST scenario.

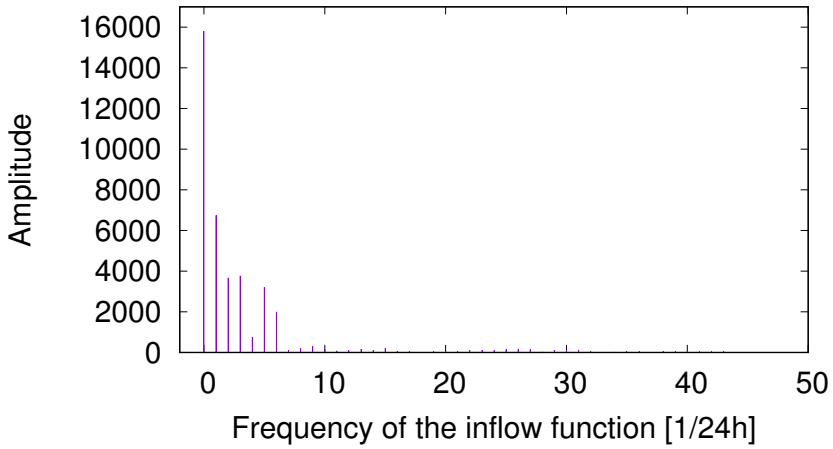


Figure 43: Discrete Fourier transform of the inflow function from the LuST scenario.

In order to pick ψ in a reasonable manner, we first have to take a look at the characteristics of realistic inflow patterns. To achieve this, we have measured the inflow of cars into the road network in the LuST scenario [28]. The LuST scenario, in essence, is a model of the city of Luxembourg which, according to the authors, resembles a standard topology that is common in mid-size European cities. It furthermore is equipped with real information concerning traffic demands and mobility patterns for a period of 24 hours [28]. Figure 41 depicts the cumulative distribution function of the departures. Furthermore, Figure 42 shows the traffic inflow function in the LuST scenario. In a next step we have computed the discrete Fourier transform (DFT) [16] of the inflow function. The results can be seen in Figure 43. In this context, the x axis represents the frequency of a variable cosine inflow function and the y axis the according amplitude. From this realistic scenario, hence, we learn that the maximum number of full periods in the variable inflow function is 6. To put it differently, this shows us that the inflow function for realistic traffic patterns does not fluctuate more than 6 times during the entire simulation run. For our cosine function that would been, that we have at most $\psi \leqslant 6$ full periods during our observation time frame T.

Parameter ΔQ is the amplitude of the variable inflow function and has to be defined prior to an evaluation run. Assume that τ_{max} is the maximum possible inflow that a particular scenario can handle and $\tau_{freeflow}$ denotes the maximum possible inflow rate that, when applied, allows all cars to travel at free flow speed. These two values have to be determined empirically for complex microscopic scenarios. After all, we suggest choosing ΔQ so that:

$$Q + \Delta Q \leqslant \min(\tau_{max}, \tau_{freeflow}) \tag{16}$$

Depending on the simulator used, $\tau_{freeflow} \leqslant \tau_{max}$ does not necessarily have to hold. This can happen, for example, when the granularity of the underlying time steps is too large so that newly added cars do not move away quickly enough to open up the space for their successors.

5.6.2 *Adjusting Given SUMO Scenarios*

The challenge now is to modify existing SUMO scenarios in a way that the inflow matches the shape described by $g(t)$. Before we see how to do this, it is worth taking a quick look at how scenarios (or more importantly their traffic demand) are modeled in SUMO.

As depicted in Figure 44 the most basic way to model the traffic demand in SUMO is to provide one entry for each car in the XML formatted route file. In this context, each car is assigned with an identifier, a departure time, and a route in form of a list of edges constituting the car's planned route. As this is the most common format that

Figure 44: Illustration of XML formatted route configuration file for SUMO.

```
<routes>
   <vehicle depart="0" id="Audinot_7_0"><route edges="a131 ... a
       209 "/></vehicle>
   <vehicle depart="1" id="Borgo_100_0"><route edges="b6 b100 ...
        b3[1]b "/></vehicle>
   <vehicle depart="5" id="Certosa_9_0"><route edges="b22[0] b
       22[1] ... b2[1][1][1]b "/></vehicle>
   <vehicle depart="6" id="Costa_12_0"><route edges="a78[0] a56a
       ... b53a "/></vehicle>
   <vehicle depart="7" id="Gandhi_40_0"><route edges="b1[0] ba
       1[1] ... b53a "/></vehicle>
   <vehicle depart="8" id="Malvasia_70_0"><route edges="b52 b51
       ... a77[1][1] "/></vehicle>
   <vehicle depart="9" id="Pepoli_3_0" type="ignoring"><route
       edges="a210 a43[0] ... a14 "/></vehicle>
[...]
</routes>
```

currently available SUMO scenarios are provided in, we only focus on this case. If required, different formats—such as OD trip matrices instead of individual routes—can be transformed into this basic format using SUMO on-board tools.

In order to adapt the departure times of all cars in a way that the inflow matches the shape described by $g(t)$, it is desired to have a function $s(t) = t'$ which associates every car's original departure time t with a new departure t'. In essence, once could say that we are aiming for achieving the inflow pattern $g(t)$ by shifting the departure times of certain vehicles on a time line according to function $s(t)$ but without changing their order.

In order to design such function $s(t)$ we again take a look at Figure 40 in order to understand the following trick: we interpret that for each t_1 the function

$$N(t_1) = t_1 \cdot Q_0 \tag{17}$$

denotes the number of cars (to recall, we do not change the order of the cars) that have entered the scenario up to time t_1 in the original, untouched scenario with a steady inflow rate. The counting index is analogous to the integral: imagine that 100 cars have entered the scenario up to a certain point in time t_1. In this context, 100 can be also interpreted as the index of the car that will enter the scenario at t_1. Using the interpretation of the counting index is essential here for the next step.

More formally, $N(t_1)$ can be seen as the (counting) index of the car that will depart at time t_1 in the original scenario.

Now we define a function

$$N_g(t') = \int_0^{t'} g(t)\,dt \tag{18}$$

as the function giving back the (counting) index of the car that will depart at time t' in the variable shaped inflow scenario with inflow function $g(t)$. To once again emphasize the trick: we do not count the number of cars that have departed until a time t, but we interpret $N(t_1)$ as the index of the car (in an ordered set of cars) that will depart at time t. This allows us now to take the two functions

$$\begin{aligned} N(t_1) &= \text{index of departing car in original scenario at time } t_1 \\ N_g(t') &= \text{index of departing car in adjusted scenario at time } t' \end{aligned} \tag{19}$$

and search for a function that gives us a relation between a specific car's original departure time t_1 and its new departure time t', i.e., we are looking to a function $s(t_1)$ which maps a car's original departure time t_1 to its new departure time $s(t_1) = t'$ so that $N_g(s(t_1)) = N(t_1)$ is met. To put it differently, when $N(t_1)$ tells us that car with index i departed at t_1 we are now looking for the i-th car's new departure time $s(t_1)$ in the adjusted variable-inflow scenario for which $N_g(s(t_1))$ must, by definition, return the very same car index i.

This essentially means, we are required to solve the integral equation

$$N_g(s(t_1)) = N(t_1) \Leftrightarrow \int_0^{s(t_1)} g(t)\,dt = t_1 \cdot Q_0 \tag{20}$$

With the substitution $s(t_1) \mapsto u \Leftrightarrow t_1 \mapsto s^{-1}(u)$ we can write

$$\int_0^{u} g(t)\,dt = s^{-1}(u) \cdot Q_0 \tag{21}$$

and solve for

$$s^{-1}(u) = \frac{1}{Q_0} \cdot \int_0^{u} g(t)\,dt \tag{22}$$

Inserting (14) for $g(t)$ into (21) gives us

$$\int_0^{u} Q_0 - \Delta Q \cdot \cos(t \cdot \frac{\psi \cdot 2 \cdot \pi}{T})\,dt = u \cdot Q_0 - \frac{\Delta Q \cdot T}{\psi \cdot 2 \cdot \pi} \cdot \sin(u \cdot \frac{\psi \cdot 2 \cdot \pi}{T}) \tag{23}$$

and with the former substitution $t_1 = s^{-1}(u)$

$$t_1 = s^{-1}(u) = u - \frac{\Delta Q}{Q_0} \cdot \frac{T}{\psi \cdot 2 \cdot \pi} \cdot \sin(u \cdot \frac{\psi \cdot 2 \cdot \pi}{T}) \tag{24}$$

Currently, we only have $t_1 = s^{-1}(u)$ which would give us the original time t_1 for any time u that a car departed at in the adjusted scenario using $g(t)$ as the inflow function. In fact, we are interested

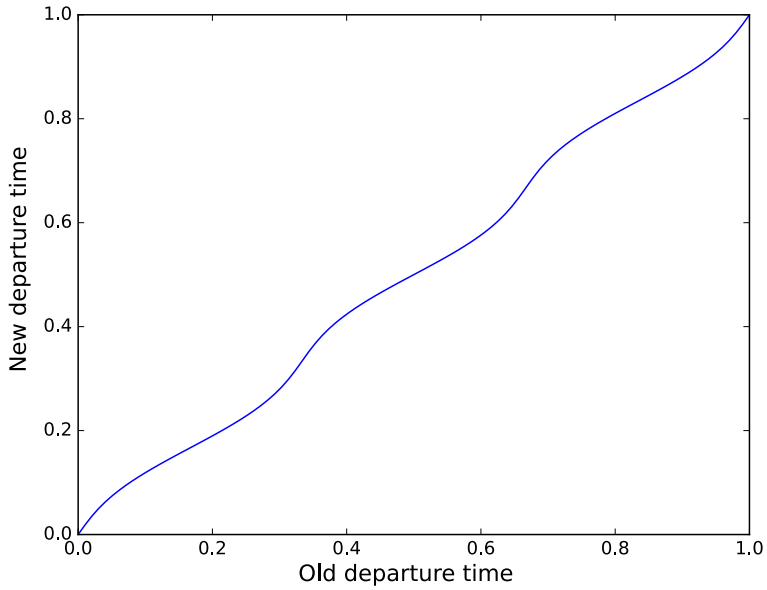


Figure 45: Variable inflow.

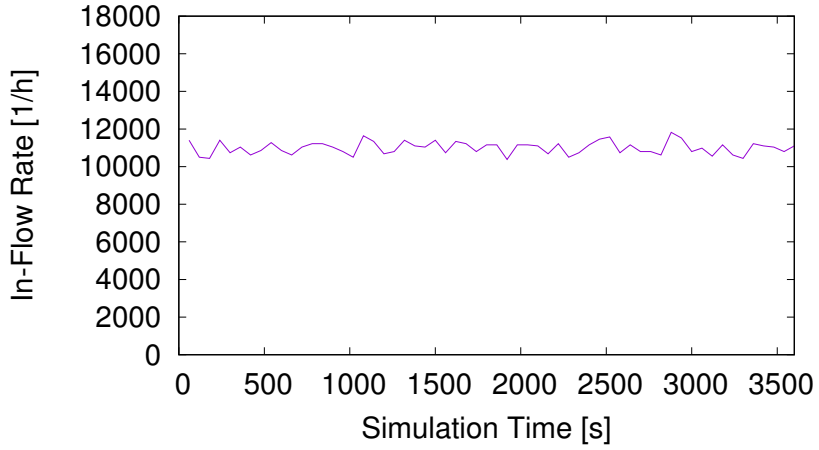
in the inverse, namely $s(t_1) = u$. As this equation cannot be analytically solved for s, we suggest using numerical methods for that.

In Figure 45 we can exemplarily see a plot showing the old departure time on the x axis and the modified departure time for the variable inflow scenario on the y axis for the time frame $0 \dots 1.0 \cdot T$, a steady inflow rate of $Q_0 = 11000$, a frequency of $f_{freq} = \frac{5}{T}$ (i.e., $\psi = 5$) and a $\Delta Q = 5000$. This ΔQ has been evaluated experimentally for the Bologna scenario from Section 5.4. It can be seen that the departure times are periodically stretched and compressed, which causes the very exact periodic inflow function $g(x)$ as seen in Figure 40.

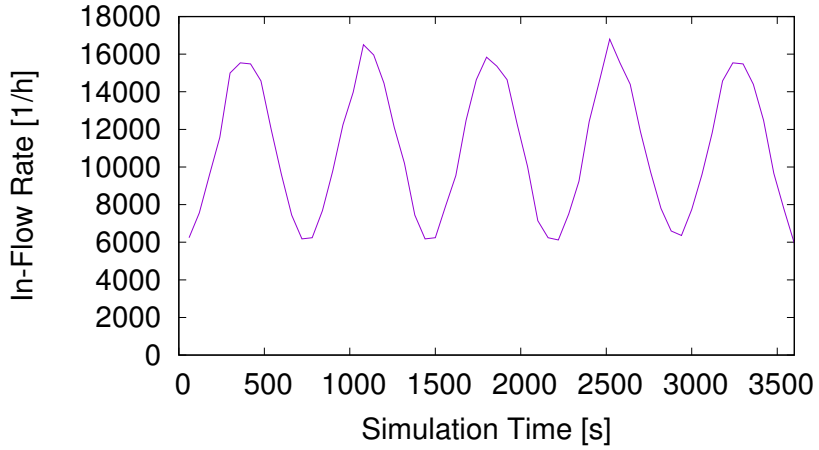
In order to show and verify the applicability of this methodology to actual microscopic simulations, where we surely do not have a perfectly constant inflow, we have applied the flow-adjusting method to the Bologna. The parameters for Q_0, ΔQ and ψ remained unchanged. We have plotted the measured (not calculated) inflows of the original, untouched scenario in Figure 46a as well as for the modified scenario in Figure 46b. The applicability of our methodology can be seen as confirmed as the pattern in the modified scenario looks as desired.

5.6.3 *Impact of Variable Inflow on the Optimization*

Earlier, we have shown that the proposed distributed GA approach does achieve good results. However, we have tested it only in scenarios with a constant traffic inflow. In this section we want to evaluate the impact a variable inflow has on the optimization result achieved by our proposed approach.



(a) Original inflow measured by SUMO in Bologna scenario.



(b) Modified inflow measured by SUMO in Bologna scenario.

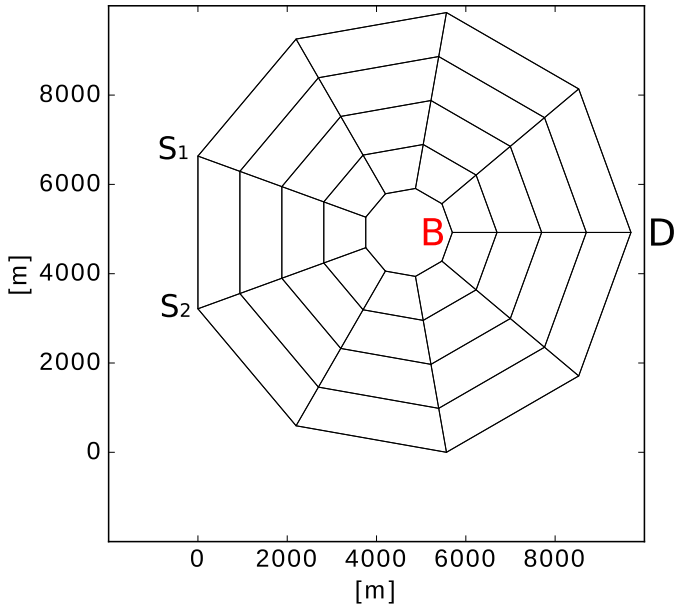


Figure 47: Example spider network.

For this experiment we decided to use an artificial spider scenario instead of conducting the evaluation in the the Bologna scenario. This choice has been made, because such artificial and simple scenarios make it easier to *understand* the effects that we can observe. This is not necessarily the case for more complex scenarios where many unknown interdependencies may occur and obscure the results. Our spider scenario is depicted in Figure 47 and consists of nine arms and five concentric layers. There are two traffic flows, i.e., a flow from S_1 to D, which takes the upper half of the center ring, and a flow from S_2 to D, which takes the lower half of the center ring. Both flows share the common bottleneck B. The scenario was calibrated with a steady inflow rate of 720 vehicles per hour. This is as the same time the maximum inflow that does not cause any congestion at the bottleneck B; any higher values do so. The total simulation time T is 10000 seconds.

Aside from the static inflow, for which we define the number of oscillations ψ to be $\psi = 0$, we have created multiple variations of this scenario using the methodology described in Section 5.6.2. In this context we have varied ψ in the range $\psi \in \{0.5, 1 \ldots 10\}$. As we have learned in Section 5.6.1 that ψ hardly exceeds the value of 6 under realistic conditions, we argue that it is sufficient to limit the evaluation to this range. The evaluation of $\psi = 0.5$, in this context, was required to reassure that the congestion for $\psi = 1$ is the "peak" or if traffic conditions worsen even more for smaller values of ψ. This was not the case here. Table 7 shows the measured average travel

Table 9: Evaluation results for different variable inflows.

Psi	One-step	DUE	Global GA	Dist GA	Behind	Ratio
0	819.95 s	819.95 s	819.95 s	819.95 s	0%	–
0.5	1633.25 s	1449.21 s	1094.22 s	1181.55 s	7.98%	0.33
1	2357.63 s	1780.08 s	1110.47 s	1271.62 s	14.51%	0.32
2	1598.94 s	1319.95 s	1089.18 s	1158.42 s	6.35%	0.43
3	1341.09 s	1154.13 s	1035.91 s	1074.93 s	3.76%	0.49
4	1234.21 s	1095.71 s	1032.90 s	1075.62 s	4.13%	2.13
5	1128.51 s	1039.90 s	991.92 s	1026.32 s	1.45%	2.53
6	1075.78 s	1009.57 s	967.91 s	984.16 s	3.46%	0.64
7	1051.81 s	984.00 s	954.32 s	966.38 s	1.26%	0.68
8	1016.80 s	969.23 s	946.13 s	956.76 s	1.12%	0.85
9	1013.76 s	954.11 s	932.14 s	939.83 s	0.82%	0.54
10	975.30 s	942.66 s	926.87 s	931.93 s	0.54%	0.47

times in the scenario with every driver taking the shortest path from his source to his destination (here, it is referred to as the *untouched* case) as well as for adjusted routes representing the dynamic user equilibrium (DUE), the approximation of the system optimum that we obtain from the global GA (in the remainder always referred to as just global GA), and routes that we have obtained using the distributed GA methodology proposed in this chapter. In the case of $\psi = 0$ every car can travel at its free flow speed and so the DUE and the global GA are equal. This is a direct result from the fact that we have a steady inflow rate which is right at the limit of what the bottleneck can handle but without exceeding it. In this case, there is no optimization potential. This, of course, looks different for other values of ψ. Looking at the travel times in the *untouched* case, it can be seen that for $\psi = 1$ the congestion in the road network is higher than for higher values of ψ. This can be explained with only one period of the cosine inflow function that causes the maximum inflow rate the network can handle without any congestion to be exceeded for a fairly long time span. This also means, that different ψ have different optimization potentials and so the difference between the Dynamic User Equilibrium and the System optimum is different.

It can be seen, that the results of the distributed GA approach in all cases lie between the DUE and the global GA, which shows that the algorithm does not produce bad results such as results that are worse than selfish routing (DUE)—for any "realistic" value of ψ. To recall, we have earlier shown that $\psi \leqslant 6$ can be assumed realistic. The column *Behind* indicates how far off our distributed GA approach is

compared to the approximation of the system optimum. It can be seen, that our distributed GA approach is 14.51% behind in the worst case, i.e., the case with the largest gap between the DUE and the global GA. In all other cases, the difference to the global GA ranges from 6.35% to 0.54%.

Furthermore, we were interested in whether the result of our optimization was closer to the SO or closer to the DUE. This can give us an insight of the overall quality of our solution. In this context we have evaluated the ratio $\frac{\alpha}{\beta}$ with α being the difference of the distributed GA approach to the SO and β being the difference of the distributed GA approach to the DUE. This result we have put into the *Ratio* column of Table 9. In this context, a ratio < 1 means that the optimization result is closer to the global GA than it is to the DUE. Except for $\psi \in \{4, 5\}$ this is the case for all evaluations.

To summarize, we can say that the distributed GA approach produces results that lie in between the global GA and the DUE for multiple variable inflow functions of different realistic frequencies. Furthermore, the results are in almost all cases closer to the global GA than they are to the DUE.

5.7 IMPROVING PRECISION WITH HISTORICAL TRAFFIC DATA

Our approach applies a distributed GA to dynamically changing traffic situations. In this context it continuously adapts the route choices of all cars that are in the scenario and that are equipped with a device running our proposed algorithm so that they are desirably optimal for the current traffic demand. That means, that the algorithm can only adapt to changing traffic conditions in a way, that it finds desirably optimal partial routes from where the cars' are positioned currently to their desired destinations. The clear disadvantage of this approach is that, if at some point in time it becomes clear that taking a different route in the past would have been better, it is impossible to go back in time and make that decision. In fact, cars that enter the scenario in the future and that are not known at the time the GA outputs a solution can cause that the result is—in the worst case—far away from a global optimal solution.

To this end it seems natural to think about ways to minimize this impact so far as it is possible. In fact, if we could a-priori predict accurately which cars will appear when in the future as well as their source and their destination, we could approximate a global optimum solution assuming a 100% penetration rate. While both the assumption of perfect future knowledge as well as a 100% penetration rate are unrealistic, we propose the following method to at least move into that direction as far as the accuracy of our prediction of the future allows us to.

The overall goal is to allow for a time-dependent prediction of future traffic and use that prediction in our genetic algorithm. To achieve this, we first assume that we have a known "daily pattern" which is given in the form of a time-dependent OD matrix. Time-dependent OD matrices can be, e.g., obtained by methods discussed in [3, 4, 113] Each entry of this matrix represents the volume of traffic that, in a time interval h, goes from origin O to destination D.

Without the loss of generality, we assume that we are currently at time t_0 and the local knowledge sets of all cars that actually *are* in the scenario are filled completely. Now, when a traffic simulation is being performed based on the information in the knowledge sets in order so evaluate the fitness value, it is equipped with so-called *ghost cars*. Ghost cars are, and this is the key idea here, cars that have a departure time $t > t_0$, i.e., cars that have their departure time in the future and thus are not included neither in any knowledge set nor in the road network itself. Nevertheless, they are important as the simulation is required to account for the future traffic situation which of course is largely affected by cars with a future departure time. To put it differently, imagine we would simulate traffic only with those cars that are currently in the road network. This, again, would mean that we have a network with cars flowing out of the network but no new cars entering the network which constitutes a network with continuously decreasing traffic load. This, however, does not reflect the reality. In fact, cars are expected to enter the network in the future. These ghost cars, therefore, are added to the network as a filler for the uncertain future and simulate a probable future traffic influx based on an empirically observed daily pattern. To point out, these ghost cars are just added to the simulation and not to the chromosome. That is, they are not part of the optimization problem itself.

5.8 INFLUENCE OF DIFFERENT COMPUTATION SPEEDS

The above evaluation was performed as a so-called discrete event simulation (DES) [39]. Generally speaking, DES correspond to the simulation of some system—here a road traffic scenario—which evolves through time. More precisely, it is assumed that a system changes its state at discrete points in simulation time. Such discrete points can describe, e. g., the time when a beacon is received, when a driver makes a decision, when the GA instance finishes computing yet another generation, etc. Those events are scheduled on a virtual time line which is entirely separated from the real time line. That is, typically, events that need a lot of calculation time in reality, happen instantly at their scheduled time on the virtual clock. This behavior is not wished for in most cases. For example, if the transmission of a large file over a TCP connection is modeled in a DES, it is required to actually schedule two events. On the one hand, the event that sends the file itself, and

Table 11: Impact of different generation durations on the quality of the result.

Generation duration	Average travel time	Gain
128s	290.25 s	4.35%
64s	281.32 s	7.29%
32s	288.24 s	5.01%
16s	270.22 s	10.95%
8s	256.74 s	15.39%
4s	249.94 s	17.63%
2s	244.01 s	19.59%
1s	242.28 s	20.16%
0.5s	247.11 s	18.57%
0.25s	239.06 s	21.22%
0.125s	241.42 s	20.44%

on the other hand the event which indicates that the file has been received on the receiver side. The time between these two events (again, on the virtual time line) can be modeled arbitrarily. A larger time span means a slower transmission time and vice versa. The same principle applies to the methodology that we have suggested above: The calculation of one generation of the GA instance that is running locally in the computation nodes is in fact a composition of two events inside our DES. That is, the start of the calculation and the end of the calculation model. in the above evaluation we have assumed that the calculation of one generation of a GA instance takes approximately one second. This value has been evaluated empirically. To recall, as described in Section 5.4, one generation requires to conduct 20 individual traffic simulations, one for each individual in the population. According to [111] a modern portable navigation system like, e. g., the TomTom GO 6100 comes equipped with a 600 MHz single core processor. As these navigation systems usually do not permit to execute own software on it, we had to find something that offers a similar CPU speed. According to [110] the Raspberry Pi (Model A) offers a 700 MHz single core CPU. Therefore, we assumed that the Raspberry Pi could give us a pretty good indication of how performant our proposed approach would run on an actual navigation device. We have found that the calculation of one generation, i.e., the calculation of 20 individual traffic simulations of the aforementioned Bologna scenario, took 1s on the Raspberry Pi. This, to point it out, is also the required time frame for one generation that we have assumed in our former evaluation.

However, it is interesting to evaluate how the performance of our proposed distributed GA approach is affected by different CPU speeds and, as a matter of fact, by different durations for the calculation of one GA generation. Table 11 shows the overall performance for a number of different durations. It can be clearly seen that the quality of the optimization result does not change much for computation times shorter than 4s. Then, the achievable gain drops quickly. For a generation duration of 128s we can only reach a gain of 4.35%. While this value is still comparable to the FCD on central server approach both distributed and with a 360s lag, it is still fairly bad.

The stability of the achieved gain for generation times lower than 4s can be explained with the low number of generations that are required to adapt the GA instance to the changing conditions after it has approached the optimum closely enough. In this context, we could show that it takes on average 5 generations to adapt to a change in the local view of the road traffic network. To put it differently, the algorithm functions properly for all deviations from the current route that have to be taken more than $5 \cdot \text{generation} \cdot \frac{1\text{s}}{\text{generation}} = 5\text{s}$ in the future. All other deviations from the original route would be found too late, i. e., when the car has already passed the point where it should take the detour. The longer the generation duration, the less routes (that the car still can take *after* the solution was found) are in the actual search space. This also explains the significant drop in the achievable gain for generation durations up to 128s.

5.9 CHAPTER SUMMARY

The importance of an intelligent route assignment methodology was noticed by numerous previous works. The here proposed approach starts from a local view on the conditions in the road network and uses an Island GA model to find good route assignments. Each car participates in the optimization by running a local GA instance and sharing good solution candidates with other, nearby cars. As urban traffic has highly dynamic characteristics, we had to solve the problem of imperfect knowledge in the local GA instances.

We evaluated our proposed approach using a traffic model of the city of Bologna, which reflects a realistic traffic demand during one observed peak hour. As a reference, we furthermore implemented a number of alternative approaches, including route choice methods as they are found in deployed solutions today. The results show that our approach results in very favorable route choices based on a fully decentralized, cooperative optimization process both in the case of a static and a variable traffic demand. More precisely, we have shown that the travel time for the whole system is, in all cases, between the UE and the SO: this alone makes this approach superior to other approaches.

Also, we have quantified the impact on both the CO_2 (carbon dioxide) emissions as well as the fuel consumption which is caused by the route choice optimization using the online route optimization techniques described above. In this context, we evaluated the increase in emissions and fuel consumption for the new (optimized) route choices per kilometer of the route as it was driven in the scenario with unoptimized route choices. It could be shown that the system as a whole experiences an 4.05 % increase of CO_2 emission and an increase of 4.06 % in the fuel consumption per original kilometer. At the same time, the cost (in terms of the total travel time) for the entire system was reduced by over 20 %. While from the ecological point of view the route choice optimization leads to slightly poorer results, the environmental overhead is reasonably low compared to the benefit that can be achieved by the drivers in terms of a lower travel time.

6 CONCLUSION

In this thesis, we have discussed the design of a distributed, intelligent traffic information system that is capable of cooperatively managing and improving the traffic flow in arbitrary road networks. More precisely, the idea was an online optimization of individual route choices in a road network by using a *distributed GA* (dGA) named *island model*. In our scheme each car participates in the optimization by running a local GA instance and sharing good solution candidates with other, nearby cars using inter-vehicle communication.

The development of this design yielded a number of challenging questions which we tackled in this work. In Chapter 2 we have given a general introduction to the basic concepts of road traffic as well as its simulation and optimization. Then, we investigated the applicability of GAs to the route assignment problem in Chapter 3. We have demonstrated that GAs indeed are a feasible approach for different target functions to achieve a good route assignment which is superior to other approaches presented in the literature. In this step we were interested in demonstrating the applicability itself and proposing a methodology for assessing the maximum optimization potential of arbitrary road traffic scenarios. This step was necessary to have a base line for comparison with other traffic optimization techniques. Since we were interested in the best case, we could assume perfect global knowledge and did not need to address issues which are expected in dynamic environments such as an incomplete view on the traffic situation or an unexpected behaviour of the drivers.

Before thinking about how to deal with those difficulties, it was required to address a different problem first: the time required for the large amount of simulations that a GA based approach requires is a bottleneck with state-of-the art microscopic traffic simulators such as SUMO. In order to later adapt the approach for an online traffic optimization, where time is a crucial factor for the quality of the optimization result, it was required to develop faster methods to simulate traffic: in this context it was required to reduce the level of detail, or increase the level of abstraction, while at the same time retaining enough accuracy of the prediction about the future impact of certain route assignments on the traffic efficiency. In Chapter 4, we have discussed MATSim as a possible alternative. MATSim is a leight-weight microscopic traffic simulation which has been derived from Gawron's queue model and fixes some problems such as the unfair junction problem which can be found in Gawron's original paper. Along the lines, we have shown that, although MATSim constitutes a

large speed up, the results are not entirely comparable to a detailed microscopic simulator such as, e.g., SUMO. In a next step we have identified the causes for these differences: the main issue was the macroscopic treatment of junctions in Gawron's queue model. While junctions can assign priorities to certain lanes proportionally based on their traffic load, a sufficiently complex junction logic is missing. That is, depending on whether it is a right-of-way junction or an allway-stop junction, the traffic pattern may look entirely differnt. Based on our discoveries, we have presented our own adaptation of Gawron's queue model which allows the configuration of arbitrarily complex junction logics in the road network. We have shown that the results obtained by our proposed approach are comparable to the results obtained when using SUMO as the simulation engine. Due to an efficient implementation we could also beat the performance of MATSim. After all, we have shown that our proposed simulation approach is a very good approximation of a full detail microscopic simulation.

In Chapter 5 we combined the insights from the two prior chapters and proposed an approach based on the idea of island model GAs—a variant of distributed, parallel GAs. Numerous publications have shown that island models yield to better results than running one panmictic GA instance with the same amount of computation power. Our suggested approach, given a local view on the conditions in the road network, gradually evolves a set of initially chosen route assignments for all cars in the network simultaneously using the principles behing GAs. To be more precise, each car that is equipped with a device running our proposed algorithm executes a local instance of the optimization algorithm. In this context, it continuously keeps optimizing the route choices of all cars that are included in the car's local view of the reality. The individual small GA instances executed on the cars' local navigation devices make use of the fast simulation engine that was described in Chapter 4 and which made live online optimization possible in the first place. Additionally, all separate GA instances occasionally exchange information (here, the actual chromosomes) with other cars. This process helps the cars to share the *work* that they have already done with others.

For our approach to work, multiple very complex problems had to be solved like, e. g., the situation where multiple cars have different views of the reality and so work on multiple, slightly different formulations of the optimization problem. Our evaluations include the comparison against other approaches—theoretical ones as well as those found in today's commonly used navigation system—in terms of the travel time and route length driven. To account for the bootstrapping problem, we have also analyzed the performance for different penetration rates. We have seen, that even a penetration rate as low as 10% outruns other approaches which rely on floating car data. Furthermore, we have shown that our proposed approach works for differ-

ent types of inflow functions (that is, static or periodically changing) which have to be expected in real world situations. In this context we have extracted typical inflow patterns from realistic SUMO scenarios such as the LUST scenario. Furthermore, we conducted an analysis of the change in costs in terms of a potentially higher fuel consumption and/or higher emissions. These analyses have shown that our approach does not come at significantly higher environmental costs.

BIBLIOGRAPHY

[1] C. W. Ahn and R. S. Ramakrishna. "A genetic algorithm for shortest path routing problem and the sizing of populations." In: *IEEE Trans. Evolutionary Computation* 6 (6) (2002), pp. 566–579.

[2] E. Alba and J. M. Troya. "A survey of parallel distributed genetic algorithms." In: *Complexity* 4 (4) (1999), pp. 31–52.

[3] K. Ashok. "Estimation and prediction of time-dependent origin-destination flows." PhD thesis. Massachusetts Institute of Technology, 1996.

[4] K. Ashok and M. E. Ben-Akiva. "Dynamic origin-destination matrix estimation and prediction for real-time traffic management systems." In: *ISTTT '93: Proceedings of the 12th International Symposium on the Theory of Traffic Flow and Transportation*. Berkeley, CA, USA, July 1993.

[5] T. Bäck and F. Hoffmeister. "Extended selection mechanisms in genetic algorithms." In: *ICGA '91: Proceedings of the Fourth International Conference on Genetic Algorithms*. San Diego, CA, USA, July 1991, pp. 92–99.

[6] J. E. Baker. "Reducing bias and inefficiency in the selection algorithm." In: *ICGA '87: Proceedings of the Second International Conference on Genetic Algorithms*. Cambridge, UK, July 1987.

[7] A. L. Bazzan. "Paying the Price of Learning Independently in Route Choice." In: *Progress in Artificial Intelligence*. Springer, 2013, pp. 42–53.

[8] A. L. Bazzan and F. Klügl. "Introduction to intelligent systems in traffic and transportation." In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 7 (3) (2013).

[9] M. J. Beckmann. "On knowledge networks in science: collaboration among equals." In: *The Annals of Regional Science* 28 (3) (1994), pp. 233–242.

[10] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz. "SUMO – Simulation of Urban MObility: An Overview." In: *SIMUL '11: Proceedings of the Third International Conference on Advances in System Simulation*. Barcelona, Spain, Oct. 2011.

[11] M. Ben-Akiva, H. N. Koutsopoulos, C. Antoniou, and R. Balakrishna. "Traffic simulation with dynamit." In: *Fundamentals of Traffic Simulation*. Springer, 2010, pp. 363–398.

[12] L. Bieker, D. Krajzewicz, A. Morra, C. Michelacci, and F. Cartolano. "Traffic simulation for all: a real world traffic scenario from the city of Bologna." In: *SUMO '14: Proceedings of the Conference on Modeling Mobility with Open Data*. Berlin, Germany, May 2014, pp. 47–60.

[13] T. Blickle and L. Thiele. "A comparison of selection schemes used in evolutionary algorithms." In: *Evolutionary Computation* 4 (4) (1996), pp. 361–394.

[14] T. Börgers and R. Sarin. "Learning through reinforcement and replicator dynamics." In: *Journal of Economic Theory* 77 (1) (1997), pp. 1–14.

[15] P.-D.D. D. Braess. "Über ein Paradoxon aus der Verkehrsplanung." In: *Unternehmensforschung* 12 (1) (1968), pp. 258–268.

[16] E. O. Brigham. *The Fast Fourier Transform and Its Applications*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988. ISBN: 0-13-307505-2.

[17] L. Buriol, M. Ritt, F. Rodrigues, and G. Schäfer. "On the Smoothed price of anarchy of the Traffic Assignment Problem." In: *ATMOS '11: 11th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*. Saarbrücken, Germany, Sept. 2011, p. 122.

[18] L. S. Buriol, M. J. Hirsch, P. M. Pardalos, T. Querido, M. G. Resende, and M. Ritt. "A biased random-key genetic algorithm for road congestion minimization." In: *Optimization Letters* 4 (4) (2010), pp. 619–633.

[19] D. Cagara and B. Scheuermann. "Quantifying the Influence of Route Choice Optimization on Emissions and Fuel Consumption." In: *FG-IVC '16: 4. GI/ITG KuVS Fachgespräch Inter-Vehicle Communication*. Ulm, Germany, Mar. 2016, pp. 19–20.

[20] D. Cagara, B. Scheuermann, and A. L. Bazzan. "A Methodology to Evaluate the Optimization Potential of Co-ordinated Vehicular Route Choices." In: *FG-IVC '13: 2. GI/ITG KuVS Fachgespräch Inter-Vehicle Communication*, Innsbruck, Austria, Feb. 2013, pp. 11–14.

[21] D. Cagara, A. L. Bazzan, and B. Scheuermann. "Getting you faster to work: a genetic algorithm approach to the traffic assignment problem." In: *GECCO '14: 16th Annual Genetic and Evolutionary Computation Conference*. Vancouver, Canada, July 2014, pp. 105–106.

[22] D. Cagara, B. Scheuermann, and A. L. Bazzan. "Traffic optimization on Islands." In: *VNC '15: Proceedings of the IEEE Vehicular Networking Conference 2015*. Kyoto, Japan, Dec. 2015, pp. 175–182.

[23] E. Cantú-Paz. "A survey of parallel genetic algorithms." In: *Calculateurs paralleles, reseaux et systems repartis* 10 (2) (1998), pp. 141–171.

[24] E. Cantu-Paz. "Designing efficient and accurate parallel genetic algorithms." PhD thesis. UIUC, 1999.

[25] N. Cetin. "Large-scale parallel graph-based simulations." PhD thesis. The Pennsylvania State University, 2005.

[26] A. J. Chipperfield and P. Fleming. "Parallel genetic algorithms: A survey." In: *ACSE Research Report No. 518r* (1994).

[27] Y.-C. Chiu, J. Bottom, M. Mahut, A. Paz, R. Balakrishna, T. Waller, and J. Hicks. "Dynamic traffic assignment: A primer." In: *Transportation Research E-Circular* (E-C153) (2011).

[28] L. Codeca, R. Frank, and T. Engel. "Luxembourg SUMO Traffic (LuST) Scenario: 24 Hours of Mobility for Vehicular Networking Research." In: *Proceedings of the 7th IEEE Vehicular Networking Conference*. 2015, pp. 1–8.

[29] K. Collins and G.-M. Muntean. "TraffCon: an intelligent traffic control system for wireless vehicular networks." In: *IET '07: Proceedings of the 3rd International Conference on Intelligent Environments*. Ulm, Germany, Sept. 2007, 699–706(7).

[30] J. R. Correa and N. E. Stier-Moses. "Wardrop equilibria." In: *Wiley Encyclopedia of Operations Research and Management Science* (2011).

[31] C. F. Daganzo, N. Geroliminis, C. F. Daganzo, and N. Geroliminis. *fundamental diagram of urban traffic*.

[32] P. Davies. *Assessment of advanced technologies for relieving urban traffic congestion*. Transportation Research Board, 1991.

[33] E. W. Dijkstra. "A note on two problems in connexion with graphs." In: *Numerische mathematik* 1 (1) (1959), pp. 269–271.

[34] G. Dimitrakopoulos and P. Demestichas. "Intelligent transportation systems." In: *Vehicular Technology Magazine, IEEE* 5 (1) (2010), pp. 77–84.

[35] J. de Dios Ortúzar, L. G. Willumsen, et al. *Modelling transport*. Wiley Chichester: 2001.

[36] A. Dussutour, V. Fourcassié, D. Helbing, and J.-L. Deneubourg. "Optimal traffic organization in ants under crowded conditions." In: *Nature* 428 (6978) (2004), pp. 70–73.

[37] Á. E. Eiben, R. Hinterding, and Z. Michalewicz. "Parameter control in evolutionary algorithms." In: *IEEE Transactions on evolutionary computation* 3 (2) (1999), pp. 124–141.

[38] D. Eppstein. "Finding the k shortest paths." In: *SIAM Journal on computing* 28 (2) (1998), pp. 652–673.

[39] G. S. Fishman. *Principles of discrete event simulation.* John Wiley and Sons, New York, NY, 1978.

[40] M. Frank. "The Braess paradox." In: *Mathematical Programming* 20 (1) (1981), pp. 283–302.

[41] M. Frank and P. Wolfe. "An algorithm for quadratic programming." In: *Naval research logistics quarterly* 3 (1-2) (1956), pp. 95–110.

[42] S. M. Galib and I. Moser. "Road Traffic Optimisation Using an Evolutionary Game." In: *GECCO '11: 13th Annual Genetic and Evolutionary Computation Conference*. Dublin, Ireland, July 2011, pp. 519–526.

[43] R. K. Ganti, N. Pham, H. Ahmadi, S. Nangia, and T. F. Abdelzaher. "GreenGPS: a participatory sensing fuel-efficient maps application." In: *MobiSys '10: Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*. San Francisco, USA, June 2010, pp. 151–164.

[44] C. Gawron. "An iterative algorithm to determine the dynamic user equilibrium in a traffic simulation model." In: *International Journal of Modern Physics C* 9 (03) (1998), pp. 393–407.

[45] C. Gawron. "Simulation-Based Traffic Assignment – Computing User Equilibria in Large Street Networks." PhD thesis. University of Cologne, 1999.

[46] R. Gibbons. *A primer in game theory*. Harvester Wheatsheaf, 1992.

[47] D. E. Goldberg. "Genetic and evolutionary algorithms come of age." In: *Communications of the ACM* 37 (3) (1994), pp. 113–120.

[48] D. E. Goldberg and J. H. Holland. "Genetic algorithms and machine learning." In: *Machine learning* 3 (2) (1988), pp. 95–99.

[49] *Google Maps*. http://maps.google.com.

[50] P. de Haan and M. Keller. *Handbook Emission Factors (HBEFA) for Road Transport–Version 1.2*. Tech. rep. SAEFL BERN, 2004.

[51] P. E. Hart, N. J. Nilsson, and B. Raphael. "A formal basis for the heuristic determination of minimum cost paths." In: *Systems Science and Cybernetics, IEEE Transactions on* 4 (2) (1968), pp. 100–107.

[52] P.-J. He, K.-F. Ssu, and Y.-Y. Lin. "Sharing trajectories of autonomous driving vehicles to achieve time-efficient path navigation." In: *VNC '13: Proceedings of the IEEE Vehicular Networking Conference 2013*. Boston, USA, Dec. 2013, pp. 119–126.

[53] M. Held and R. M. Karp. "The traveling-salesman problem and minimum spanning trees." In: *Operations Research* 18 (6) (1970), pp. 1138–1162.

[54] *HERE: Maps for life*. https://maps.here.com/.

[55] J. H. Holland. "Genetic algorithms." In: *Scientific american* 267 (1) (1992), pp. 66–72.

[56] J. H. Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[57] O. Jahn, R. Möhring, A. Schulz, and N. Stier-Moses. *System-Optimal Routing of Traffic Flows with User Constraints in Networks with Congestion*. Working papers 4394-02. Massachusetts Institute of Technology (MIT), Sloan School of Management, 2004.

[58] C. Jassadapakorn and P. Chongstitvatana. "Self-Adaptation Mechanism to Control the Diversity of the Population in Genetic Algorithm." In: *arXiv preprint arXiv:1109.0085* (2011).

[59] D. Jiang and L. Delgrossi. "IEEE 802.11 p: Towards an international standard for wireless access in vehicular environments." In: *Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE*. IEEE. 2008, pp. 2036–2040.

[60] H. M. Kammoun, I. Kallel, A. M. Alimi, and J. Casillas. "Improvement of the road traffic management by an ant-hierarchical fuzzy system." In: *CIVTS '11: Proceedings of the IEEE Symposium on Computational Intelligence in Vehicles and Transportation Systems*. Paris, France, Apr. 2011, pp. 38–45.

[61] A. Khosroshahi, P. Keshavarzi, Z. Koozeh Kanani, and J. Sobhi. "Acquiring real time traffic information using VANET and dynamic route guidance." In: *CCIE '11: Proceedings of the2nd International Conference on Computing, Control and Industrial Engineering*. Wuhan, China, Aug. 2011, pp. 9–13.

[62] E. Köhler and M. Skutella. "Flows over time with load-dependent transit times." In: *SIAM Journal on Optimization* 15 (4) (2005), pp. 1185–1202.

[63] E. Koutsoupias and C. Papadimitriou. "Worst-case equilibria." In: *Annual Symposium on Theoretical Aspects of Computer Science*. Springer. 1999, pp. 404–413.

[64] S. Krauß. "Microscopic modeling of traffic flow: Investigation of collision free vehicle dynamics." PhD thesis. Universitat zu Koln., 1998.

[65] L. J. LeBlanc, E. K. Morlok, and W. P. Pierskalla. "An efficient approach to solving the road network equilibrium traffic assignment problem." In: *Transportation Research* 9 (5) (1975), pp. 309–318.

[66] G. E. Liepins and M. Hilliard. "Genetic algorithms: foundations and applications." In: *Annals of operations research* 21 (1) (1989), pp. 31–57.

[67] M. J. Lighthill and G. B. Whitham. "On kinematic waves. II. A theory of traffic flow on long crowded roads." In: *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*. The Royal Society. London, UK, June 1955, pp. 317–345.

[68] R. Liu, H. Liu, D. Kwak, Y. Xiang, C. Borcea, B. Nath, and L. Iftode. "Themis: A participatory navigation system for balanced traffic routing." In: *VNC '14: Proceedings of the IEEE Vehicular Networking Conference 2014*. Paderborn, Germany, Sept. 2014, pp. 159–166.

[69] I. Lubashevsky, P. Wagner, and R. Mahnke. "Rational-driver approximation in car-following theory." In: *Physical Review E* 68 (5) (2003), p. 056109.

[70] A. Mambrini and D. Sudholt. "Design and Analysis of Adaptive Migration Intervals in Parallel Evolutionary Algorithms." In: *GECCO '14: 16th Annual Genetic and Evolutionary Computation Conference*. Vancouver, Canada, July 2014, pp. 1047–1054.

[71] W. Martin, J. Lienig, and J. P. Cohoon. "C6. 3 Island (migration) models: evolutionary algorithms based on punctuated equilibria." In: *B ack et al. BFM97], Seiten C* 6 ().

[72] K. Meesublak. "A Stochastic Traffic Assignment Model for Transportation Network." In: *WiCOM '11: Proceedings of the 7th International Conference on Networking and Mobile Computing*. Wuhan, China, Sept. 2011, pp. 1–4.

[73] D. K. Merchant and G. L. Nemhauser. "A model and an algorithm for the dynamic traffic assignment problem." In: *Traffic Equilibrium Methods*. Springer, 1976, pp. 265–273.

[74] H. Muhlenbein. "Evolution in time and space-the parallel genetic algorithm." In: *FOGA '91: Proceedings of the 1st Workshop on Foundations of Genetic Algorithms*. Bloomington Campus, USA, July 1991, pp. 316–337.

[75] H. Mühlenbein and D. Schlierkamp-Voosen. "Predictive Models for the Breeder Genetic Algorithm, I: Continuous Parameter Optimization." In: *Evolutionary Computation* 1 (1) (1993), pp. 25–49.

[76] *(M)ulti(A)gent (T)ransport (SIM)ulation, accessed 2008.* `http://www.matsim.com`.

[77] H. Noori and M. Valkama. "Impact of VANET-based V2X communication using IEEE 802.11p on reducing vehicles traveling time in realistic large scale urban area." In: *ICCVE '13: Proceedings of International Conference on Connected Vehicles and Expo*. Las Vegas, USA, Dec. 2013, pp. 654–661.

[78] G. Ochoa, I. Harvey, and H. Buxton. "Optimal mutation rates and selection pressure in genetic algorithms." In: *GECCO '00: 2th Annual Genetic and Evolutionary Computation Conference*. Las Vegas, USA, July 2000, pp. 315–322.

[79] B. Oh, Y. Na, J. Yang, S. Park, J. Nang, and J. Kim. "Genetic Algorithm-based Dynamic Vehicle Route Search using Car-to-Car Communication." In: *Advances in Electrical and Computer Engineering* 10 (4) (2010), pp. 81–86.

[80] A. Orda and R. Rom. "Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length." In: *Journal of the ACM (JACM)* 37 (3) (1990), pp. 607–625.

[81] J Pan, M. A. Khan, I. S. Popa, K. Zeitouni, and C. Borcea. "Proactive vehicle re-routing strategies for congestion avoidance." In: *DCOSS '12: Proceedings of the 8th International IEEE Conference on Distributed Computing in Sensor Systems*. Hangzhou, China, May 2012, pp. 265–272.

[82] J. Pan, I. Popa, K. Zeitouni, and C. Borcea. "Proactive Vehicular Traffic Rerouting for Lower Travel Time." In: *IEEE Transactions on Vehicular Technology* 62 (8) (2013), pp. 3551–3568.

[83] H. Pohlheim. "Ein genetischer algorithmus mit mehrfachpopulationen zur numerischen optimierung." In: *at-Automatisierungstechnik* 43 (3) (1995), pp. 127–135.

[84] T. S. Rappaport. *Wireless communications - principles and practice*. Prentice Hall, 1996. ISBN: 978-0-13-375536-7.

[85] P. I. Richards. "Shock waves on the highway." In: *Operations research* 4 (1) (1956), pp. 42–51.

[86] M. Rondinone, J. Maneros, D. Krajzewicz, R. Bauza, P. Cataldi, F. Hrizi, J. Gozalvez, V. Kumar, M. Röckl, L. Lin, et al. "ITETRIS: a modular simulation platform for the large scale evaluation of cooperative ITS applications." In: *Simulation Modelling Practice and Theory* 34 (2013), pp. 99–125.

[87] T. Roughgarden. *Selfish Routing and the price of anarchy*. MIT Press, 2005.

[88] T. Roughgarden. "The price of anarchy is independent of the network topology." In: *Journal of Computer and System Sciences* 67 (2) (Sept. 2003), pp. 341–364. ISSN: 0022-0000. DOI: 10.1016/S0022-0000(03)00044-8.

[89] J. Rybicki, B. Scheuermann, W. Kiess, C. Lochert, P. Fallahi, and M. Mauve. "Challenge: Peers on Wheels – A Road to New Traffic Information Systems." In: *MobiCom '07: Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking*. Montreal, QC, Canada, Sept. 2007, pp. 215–221.

[90] J. Rybicki, B. Scheuermann, M. Koegel, and M. Mauve. "PeerTIS: A Peer-to-peer Traffic Information System." In: *VANET '09: Proceedings of the 6th ACM International Workshop on VehiculAr Inter-NETworking*. Beijing, China, Sept. 2009, pp. 23–32.

[91] A. Sadek, B. Smith, and M. Demetsky. "Dynamic traffic assignment: Genetic algorithms approach." In: *Transportation Research Record: Journal of the Transportation Research Board* (1588) (1997), pp. 95–103.

[92] D. Schrank, T. Lomax, and S. Turner. "TTI's 2010 urban mobility report powered by INRIX traffic data." In: *Texas Transportation Institute, The Texas A&M University System* 17 (2010).

[93] D. Schrank, B. Eisele, T. Lomax, and J. Bak. *Urban Mobility Scorecard*. Tech. rep. Technical Report August, Texas A&M Transportation Institute and INRIX, Inc, 2015.

[94] W. Sha, D. Kwak, B. Nath, and L. Iftode. "Social vehicle navigation: integrating shared driving experience into vehicle navigation." In: *HotMobile '13: Proceedings of the 14th Workshop on Mobile Computing Systems and Applications*. Jekyll Island, USA, Feb. 2013, pp. 16–21.

[95] Y. Shigehiro, T. Miyakawa, and T. Masuda. "Road Traffic Control Based on Genetic Algorithm for Reducing Traffic Congestion." In: *IEEJ Transactions on Electronics, Information and Systems* 131 (2011), pp. 1190–1198.

[96] P. Spiessens and B. Manderick. "A massively parallel genetic algorithm: Implementation and first analysis." In: *ICGA '91: Proceedings of the Fourth International Conference on Genetic Algorithms*. San Diego, CA, USA, July 1991, pp. 279–286.

[97] S. Stanhope and J. Daida. "Optimal mutation and crossover rates for a genetic algorithm operating in a dynamic environment." In: *Evolutionary Programming VII*. Vol. 1447. Springer Berlin Heidelberg, 1998, pp. 693–702.

[98] R. Tanese. "Distributed genetic algorithms." In: *ICGA '89: Proceedings of the Third International Conference on Genetic Algorithms*. Fairfax County, Virginia, USA, Nov. 1989, pp. 434–439.

[99] M. Treiber, A. Kesting, and D. Helbing. "Delays, inaccuracies and anticipation in microscopic traffic models." In: *Physica A: Statistical Mechanics and its Applications* 360 (1) (2006), pp. 71–88.

[100] R. K. Ursem. "Diversity-guided evolutionary algorithms." In: *Parallel Problem Solving from Nature—PPSN VII*. Springer, 2002, pp. 462–471.

[101] A. Varga. "OMNeT++ http://www.omnetpp.org." In: *IEEE Network Interactive* 16 (4) (2002).

[102] *VEINS*. http://veins.car2x.org.

[103] J. G. Wardrop. "ROAD PAPER. SOME THEORETICAL ASPECTS OF ROAD TRAFFIC RESEARCH." In: *ICE Proceedings: Engineering Divisions*. Vol. 1. 3. Ice Virtual Library. 1952, pp. 325–362.

[104] J. G. Wardrop and J. Whitehead. "Correspondence. Some Theoretical Aspects of Road Traffic Research." In: *ICE '52: Proceedings of the Institution of Civil Engineers*. May 1952, pp. 767–768.

[105] *WAZE Navigation*. http://www.waze.com.

[106] K. P. Weinfurt. "Repeated measures analysis: ANOVA, MANOVA, and HLM." In: *Reading and Understanding More Multivariate Statistics* (2000).

[107] D. Whitley and T. Starkweather. "Genitor II: A distributed genetic algorithm." In: *Journal of Experimental & Theoretical Artificial Intelligence* 2 (3) (1990), pp. 189–214.

[108] D. Whitley, S. Rana, and R. B. Heckendorn. "The island model genetic algorithm: On separability, population size and convergence." In: *Journal of Computing and Information Technology* 7 (1999), pp. 33–48.

[109] L. D. Whitley et al. "The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best." In: *ICGA '91: Proceedings of the Fourth International Conference on Genetic Algorithms*. San Diego, CA, USA, July 1991, pp. 116–123.

[110] Wikipedia. *Raspberry Pi — Wikipedia, The Free Encyclopedia*. [Online; accessed 05-June-2016]. 2016. URL: `\url{https://de.wikipedia.org/wiki/Raspberry_Pi}`.

[111] Wikipedia. *TomTom — Wikipedia, The Free Encyclopedia*. [Online; accessed 05-June-2016]. 2016. URL: `\url{https://de.wikipedia.org/wiki/TomTom}`.

[112] D. Wilkie, J. P. van den Berg, M. C. Lin, and D. Manocha. "Self-Aware Traffic Route Planning." In: *AAAI '11: Proceesings of the 25th Conference on Artificial Intelligence*. San Francisco, USA, Aug. 2011, pp. 1521–1527.

[113] L. Willumsen. "Estimating time-dependent trip matrices from traffic counts." In: *ISTTT '84: Proceedings of the 9th International Symposium on the Theory of Traffic Flow and Transportation*. Delft, Netherlands, July 1984, pp. 397–411.

[114] T. Yamashita, K. Izumi, and K. Kurumatani. "Car navigation with route information sharing for improvement of traffic efficiency." In: *ITSC '04: Proceedings of the 7th International IEEE Conference on Intelligent Transportation Systems*. Washington, DC, USA, Oct. 2004, pp. 465–470.

[115] J. Y. Yen. "Finding the k shortest loopless paths in a network." In: *management Science* 17 (11) (1971), pp. 712–716.

[116] S. Yousefi, M. S. Mousavi, and M. Fathy. "Vehicular ad hoc networks (VANETs): challenges and perspectives." In: *ITS Telecommunications Proceedings, 2006 6th International Conference on*. IEEE. 2006, pp. 761–766.

[117] J. Zhang, F.-Y. Wang, K. Wang, W.-H. Lin, X. Xu, and C. Chen. "Data-driven intelligent transportation systems: A survey." In: *Intelligent Transportation Systems, IEEE Transactions on* 12 (4) (2011), pp. 1624–1639.

[118] G. Zoutendijk. *Methods of feasible directions: a study in linear and non-linear programming*. Elsevier, 1960.